VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Mobilní aplikace pro správu projektů

Mobile Application for Project Management

2020 Roman Šubert

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Roman Šubert**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Mobilní aplikace pro správu projektů
Mobile Application for Project Management

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit mobilní aplikaci na platformě Xamarin Forms, která bude sloužit pro jednoduchou správu projektů (vytváření projektů, uživatelů, úkolů, chyb, komentářů, nahrávání souborů, atd…). Mobilní aplikace bude fungovat jako tlustý klient pro webovou aplikaci pro správu projektů (tvorba webové aplikace není součástí této práce). Aplikace bude dále schopna přijímat upozornění na různé akce v systému (například vytvoření úkolu, či změna stavu).

Hlavní body zadání:
1. Přehled a srovnání technologií pro tvorbu multiplatformních mobilních aplikací.
2. Návrh a implementace mobilní aplikace.
3. Shrnutí dosažených výsledků.

Seznam doporučené odborné literatury:

[1] HERMES, Dan. Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. California: Apress, 2015. ISBN 978-1-4842-0214-2.
[2] VERSLUIS, Gerald. Xamarin.forms essentials. New York, NY: Springer Science+Business Media, 2017. ISBN 978-1-4842-3239-2.
[3] NEIL, Theresa. Mobile design pattern gallery. Sebastopol, CA: O'Reilly, c2012. ISBN 978-1449314323.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2020 .....

Rád bych na tomto místě poděkoval Ing. Janu Janouškovi za odbornou pomoc a rady při vedení bakalářské práce.

Abstrakt

Cílem bakalářské práce je srovnání technologií pro tvorbu multiplatformních aplikací se zaměřením především na framework Xamarin.Forms, ve kterém se bude vyvíjet mobilní aplikace pro správu projektů. Tato aplikace bude obsahovat klasický systém na vytváření uživatelů, projektů, příspěvků a úkolů. Dále bude schopna přijímat a zasílat upozornění na uskutečněné akce. Systém je doplněn rolemi s daným rozsahem, na jehož základě se uživatelům přidělují dodatečná práva, umožňující vést a dohlížet na běžné uživatele při práci s aplikací. V závěru práce je popsáno provedené otestování uživateli s pevně zadanými úkoly a jejich vyhodnocení.

Klíčová slova: multiplatformní vývoj, Xamarin, Xamarin.Forms, React Native, PhoneGap, NativeScript, Flutter, upozornění, mobilní aplikace

Abstract

The aim of the bachelor thesis is the comparison of technologies for creating cross-platform applications, focusing primarily on Xamarin.Forms framework in which mobile application for project management will be developed. This application will contain a classic system for creating users, projects, posts and tasks. It will also be able to receive and send notifications of performed actions. The system is supplemented by roles with a given scope, on the basis of which users are assigned additional rights, enabling them to guide and oversee ordinary users when working with the application. At the end of the work, the performed user testing with fixed tasks is described and evaluated.

Keywords: cross-platform development, Xamarin, Xamarin.Forms, React Native, PhoneGap, NativeScript, Flutter, notifications, mobile application

Obsah

Seznam použitých zkratek a symbolů

API	–	Application programming interface
XML	–	eXtensible Markup Language
JSON	–	JavaScript Object Notation
HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheet
XAML	–	Extensible Application Markup Language
GPS	–	Geolocation Positioning System
URL	–	Uniform Resource Locator
ANH	–	Azure Notification Hubs
FCM	–	Firebase Cloud Messaging
APNs	–	Apple Push Notification service
SSL	–	Secure Sockets Layer
IDE	–	Integrated Development Environment
APK	–	Android Application Package

Seznam obrázků

1 Úvod

Po dlouhou dobu byly mobilní telefony určeny výhradně k volání a psaní SMS zpráv. Fungovaly tedy jako přenosné verze pevných linek, které měly navíc výhodu v tom, že obsahovaly databáze kontaktů a měly možnost posílat pouze textové zprávy. Se stále zrychlujícím se technologickým vývojem začaly být telefony mnohem „chytřejší“, lehčí a s nástupem sériové výroby také cenově dostupnější. Díky tomuto vývoji se dostaly mezi širokou veřejnost spolu s upuštěním od výhradně hlasově zaměřených aplikací.

V dnešní době je velmi častá a preferovaná skupinová práce, především při nutnosti řešit náročnější a obsáhlejší úkoly. Valná většina studentů má chytré telefony a nejinak je tomu i u mnoha zaměstnanců. Vezmeme-li v úvahu tato fakta, je více než logické vytvoření mobilní aplikace se zaměřením na správu jednotlivých projektů. Ta by fungovala v rámci daného prostředí, například právě pro studenty, zaměstnance firmy nebo různé vývojářské týmy.

Cílem této práce je společný postup při práci na větších projektech a úkolech, umožnění jejich rozložení na menší části anebo přidělení jednotlivým zaměstnancům a snazší sledování a vyhodnocování jejich práce. Musí se však rovněž brát v úvahu fakt, že nejdominantnějším operačním systémem na trhu je Android, který má spoustu odlišných hardware variací či verzí operačního systému. Jeho konkurentem se stále velkým podílem je iOS od společnosti Apple a málokdy se najde čistě homogenní skupina, kde by úplně všichni měli buď iOS, či Android.

Aby se nemusela vyvíjet dvakrát stejná aplikace nativně, je vhodnější zvolit možnost vývoje multiplatformní aplikace, která umožní přenositelnost drtivé většiny napsaného kódu.

Teoretická část bakalářské práce je rozdělena na čtyři části. První se věnuje v krátkosti historickému vývoji na poli multiplatformních aplikací. Druhá je zaměřena na představení společných výhod a nevýhod multiplatformního vývoje. Ve třetí jsou rozebírány jednotlivé frameworky, včetně příkladů jejich kódu. Závěrečná část je věnována detailnějšímu popisu platformy Xamarin a Xamarin.Forms, v nichž byla aplikace napsaná.

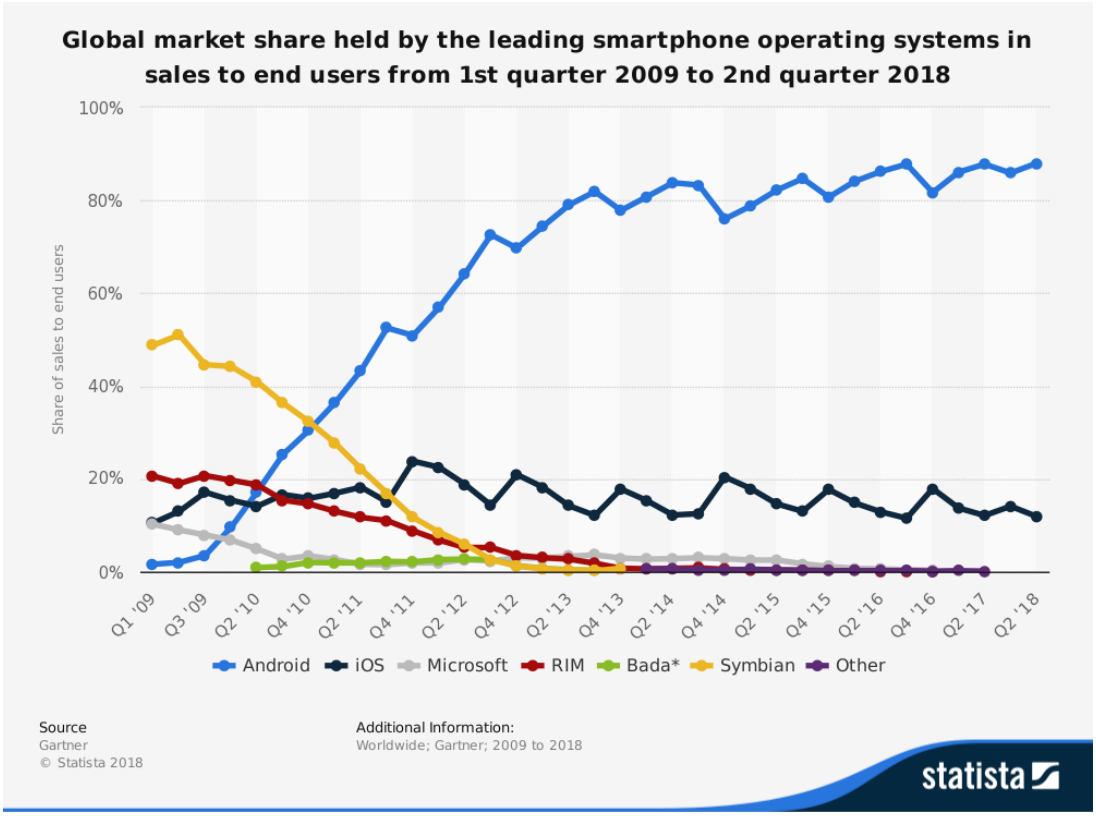
Praktická část bude začínat seznámením s návrhem včetně použitých prvků a poté bude pojednávat o implementaci tohoto návrhu.

2 Historický vývoj

V době vzniku chytrých mobilních telefonů jednoznačně dominoval iPhone od firmy Apple, s jehož příchodem, od roku 2008, můžeme poprvé hovořit o počátku éry smartphonů. Jejich budoucímu konkurentovi, který přišel s operačním systémem Android, trval příchod na trh více než rok. Zpoždění bylo způsobeno nutností reagovat na dříve vydaný iPhone. V témže roce navíc začal svou existenci Apple App Store, který umožnil pro veřejnost vyvíjet a publikovat nativní aplikace pro jejich mobilní zařízení, byť s výraznými omezeními. To jenom zvýšilo jeho popularitu a na určitou dobu posílilo dominantní postavení na trhu.

Vzhledem k neustále rostoucímu tržnímu podílu operačních systémů Android a iOS, viditelném na obrázku 1, jejichž konkurence napomohla k překotnému rozvoji mobilních aplikací, dochází k navýšení zájmu o vývoj přenositelného softwaru. Situaci navíc značně ovlivňuje fakt, že Android, který se čím dál znatelněji dostává před svého konkurenta, vykazuje značné rozdíly mezi jednotlivými produkty. Ty se týkají například hardware telefonu nebo odlišné verze operačního systému.

Nyní si již vývojáři a studia nemohou dovolit ignorovat ani jedno z těchto lukrativních odbytišť jejich produktů a kvůli rozdílnosti obou platform je nutné psát celý kód z větší části znovu od začátku. To přináší potřebu existence dvou týmů, které vyvíjejí nativně stejnou aplikaci pro iOS i Android. Takový přístup sice umožňuje na maximum využít všechny možnosti zvolené platformy, je ale vysoce neefektivní z pohledu množství času a financí potřebných pro tento vývoj. Z tohoto důvodu došlo ke vzniku a rozsáhlému rozvoji v oblasti přenositelnosti kódu a psaní multiplatformně založených aplikací.



Obrázek 1: Vývoj tržního podílu Androidu a iOS [1]

3 Výhody a nevýhody multiplatformního vývoje

Než se začneme přímo věnovat existujícím frameworkům, je třeba se zaměřit na jednotlivé výhody a nevýhody, které samotný vývoj pro vícero platforem přináší. Jedná se tedy o vlastnosti, jež jsou pro všechny stejné a plynou z rozdílného přístupu oproti nativnímu vývoji.

3.1 Výhody

Základem je existence jednotného zdrojového kódu, z něhož vycházejí tyto výhody.

Rychlejší vývoj

Řešení není třeba vyvíjet paralelně či postupně – například pro Android a iOS. Jakmile je kód napsaný, funguje na vícero platformách, což ušetří mnoho času.

Nižší náklady

Z velké části souvisejí s předchozí výhodou, která spočívá v tom, že psaní kódu pouze jednou šetří nejen čas, ale není třeba pro práci na projektech využívat další týmy se specifikovaným zaměřením na různé systémy. Tento důvod je pro mnoho firem velmi důležitý, umožňuje totiž zajistit ziskovost u aplikací, pro něž by to nešlo klasickou cestou nativního vývoje.

Širší zaměření

Tento přístup umožňuje oslovit vyšší počet koncových zákazníků, vývojáři přitom nemusí rozhodovat, kterou z existujících platforem budou preferovat.

Snazší údržba

Ať už jde o opravení chyb a nové funkce, či obecně změny a úpravy, opět se projevuje výhoda existence jednoho zdrojového kódu, v rámci něhož dochází k těmto změnám a úpravám.

3.2 Nevýhody

Rychlejší vývoj, snížené náklady a společný kód s sebou také nutně přinášejí následující úskalí.

Horší výkon

Je způsoben nutností převádět části společného kódu a prvků do nativních na základě zvolené platformy. Nativní aplikace navíc dokážou na maximum využít potenciálu nabízeného platformou. Tato nevýhoda se markantněji začíná projevovat u větších aplikací a jsou zde rozdíly mezi jednotlivými frameworky.

Kvalita grafiky

Uživatelské rozhraní, animace a 3D efekty mají omezené možnosti využívat všechny zabudované funkce, nabízené mobilním zařízením. Tato nevýhoda způsobuje nejvíce problémů při vývoji populárních mobilních her, jež na grafice a animacích hodně staví.

Nové aktualizace

Jakmile dojde k vydání nové či úpravě stávající funkce na některé z platforem, je třeba, aby došlo k jejich zakomponování v rámci frameworku. Takto dochází k časovým prodlevám, při nichž mají nativní aplikace k těmto aktualizacím dřívější přístup.

Nativní funkce

Některé z nativních funkcí nejsou přístupné v rámci multiplatformního vývoje, s čímž také úzce souvisejí možnosti spojené s hardwarovou funkcionalitou. Například přístup ke kameře, GPS anebo notifikace, které je třeba implementovat specificky pro každou platformu, viz sekce 8.

4 Existující multiplatformní řešení

V této kapitole jsou popsány rozdílné přístupy k problematice multiplatformního vývoje a jejich nejznámější představitelé, včetně konkrétních výhod a nevýhod.

4.1 HTML 5 webové aplikace

Jedná se v podstatě o klasickou webovou stránku zobrazenou na mobilním zařízení. Jejich základem jsou HTML, Javascript a CSS, na kterých běží i desktopové webové aplikace. Nemají přístup k nativním funkcím a není třeba je instalovat, jediným požadavkem je přítomnost webového prohlížeče.

4.2 Hybridní aplikace

Podobně jako výše zmíněné webové aplikace jsou postaveny na kombinaci HTML, CSS a Javascriptu, na rozdíl od nich se však zaměřuje na spouštění ve WebView obsaženém v rámci nativního aplikačního obalu.

4.2.1 Apache Cordova

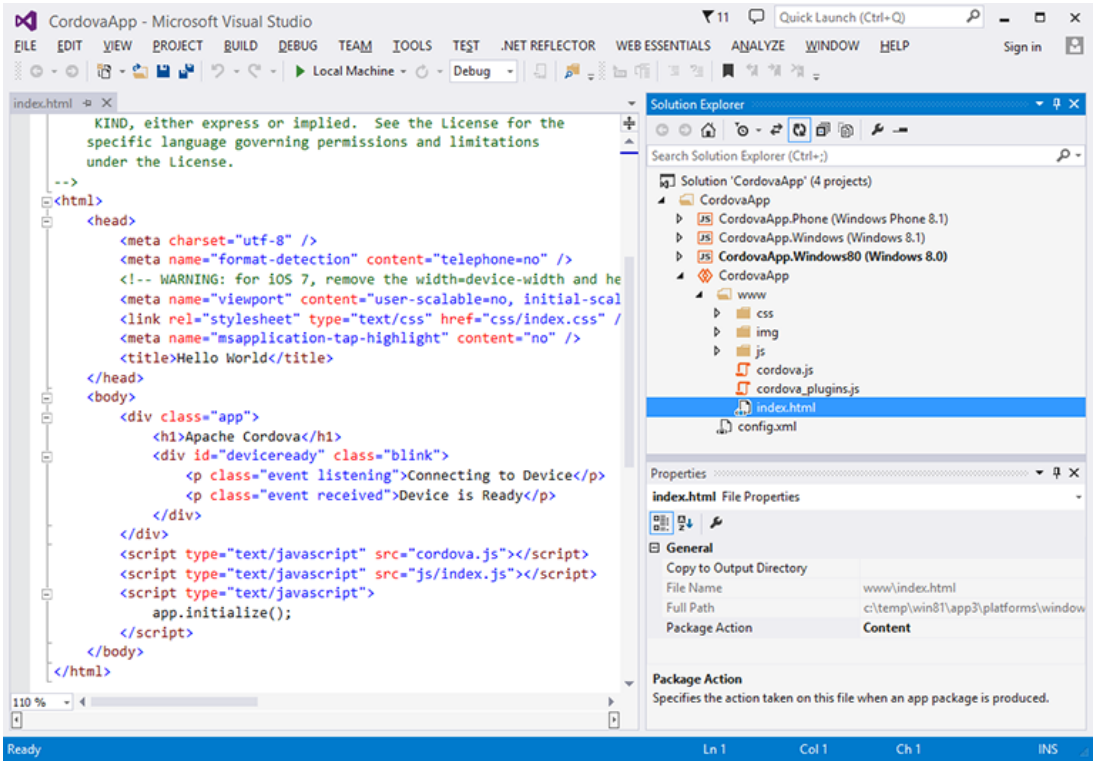
Jedná se o nejznámější veřejně dostupný framework tohoto přístupu k multiplatformnímu vývoji. Díky přítomnosti WebView lze využívat sadu javascriptových API. To jim umožňuje využívat hardwarové možnosti mobilního zařízení, jako například sensorů, dat, stavu sítě a dalších přes pluginy zabudované v nativním kódu, což je velkou výhodou oproti čistě webovým aplikacím. [2]

V souvislosti s Cordovou je určitě třeba zmínit také framework Adobe PhoneGap. Rozdíly mezi nimi jsou velmi malé, neboť Cordova v podstatě funguje jako jádro pro PhoneGap, který je tedy distribucí Cordovy, přestože se zrodila z jeho původního kódu. Především nabízí navíc služby, z nichž hlavní je PhoneGap Build Services, která se sama postará o kompilaci a umožňuje snadnější vývoj především pro operační systém iOS. [3]

S přihlédnutím k využívaným jazykům je jednoznačnou výhodou snadný přechod z vývoje webových aplikací na mobilní, není nutné se učit nové. Pro zobrazení těchto aplikací je třeba pouze webový prohlížeč. K dispozici je rovněž velké množství již existujících knihoven. Další z výhod je rozhodně snadná údržba, neboť si uživatel nemusí stahovat pokaždé novou verzi, ale jednoduše aktualizuje webové stránky. Kombinace těchto výhod pak vede k nižším nákladům a vyšší rychlosti vývoje.

Mezi nevýhody patří to, že je dostupná pouze omezená část nativní funkcionality, jejíž množství záleží na použitém frameworku. V dnešní době existuje mnoho různých prohlížečů a rovněž mobilních zařízení, které mají například odlišný hardware a rozměry obrazovky. To vyvolává nutnost dodatečné optimalizace a podporu těchto verzí. U těchto aplikací dochází k občasnému zasekávání uživatelského rozhraní, vzhledem k nadměrnému množství vrstev abstrakce.

Při práci s Cordovou je veškerý design a architektura ponechána na vývojáři, a navíc není možné vyvinutou aplikaci přímo umístit v klasickém app store, což způsobuje horší dostupnost pro koncového uživatele a s velkou pravděpodobností přispívá k nepříliš velké oblibě frameworku.



Obrázek 2: Ukázka kódu ve frameworku Apache Cordova [4]

4.3 Javascriptové frameworky

Využívají javascriptové knihovny, pomocí níž přistupují k nativním funkcím mobilního zařízení. Uživatelské rozhraní je generováno za běhu a skládá se z nativních částí.

4.3.1 React Native

Jedná se o framework, který stojí na javascriptové knihovně React, sloužící k vytváření uživatelského rozhraní. Tato knihovna, jejímž autorem byl vývojář Facebooku, je v současnosti open source. Jelikož je však React zaměřený na vývoj webových aplikací a React Native výhradně na mobilní aplikace, jsou zde samozřejmě rozdíly. Tím hlavním je fakt, že React Native neužívá Html/Css a u vzhledu se opírá výhradně o nativní API. Prozatím zůstává orientovaný pouze na platformy Android a iOS, čímž však pokrývá většinovou část operačních systémů mobilních zařízení. [5]

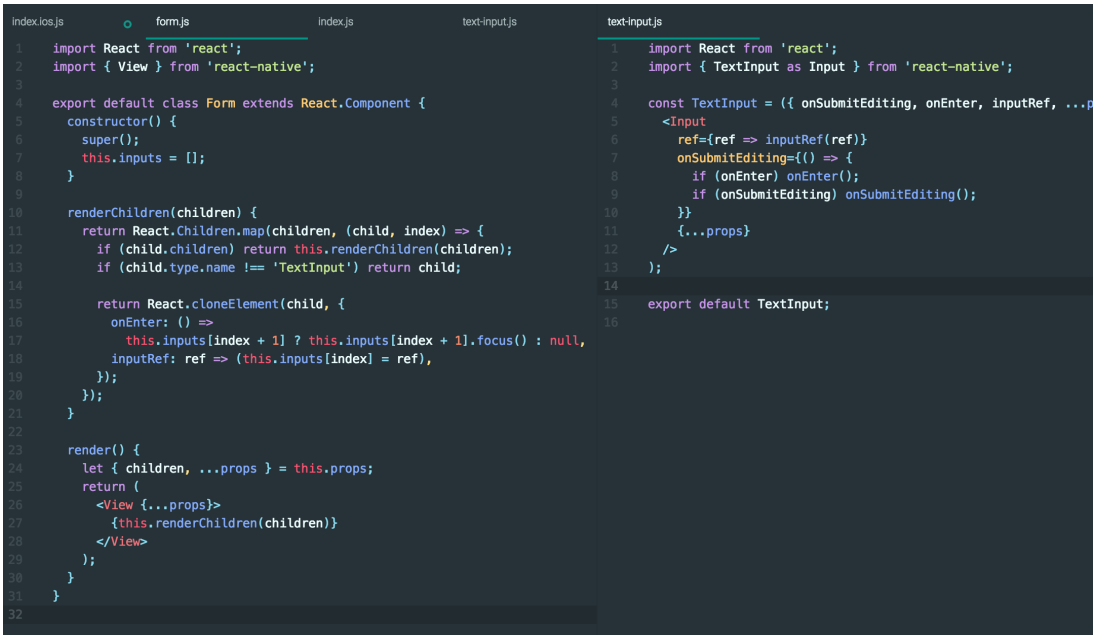
Oproti Apache Cordova nevyužívá zmíněné webviews, neboť se nejedná o prostou webovou stránku, kterou by bylo možné zobrazit přes wrapper. Naopak i přes používání kombinace

javascriptu a značkovacího jazyka podobného XML je výsledkem aplikace složená z nativních komponent platformy, pro niž je publikována.

Díky faktu, že vizuální stránka frameworku se opírá o nativní funkce a zobrazovací prvky, je intuitivní a přináší větší pružnost. Navíc veškerá logika systému neběží na hlavním jádru, čímž nedochází ke zpomalování načítání uživatelského rozhraní, a to ani v případě možných náročných algoritmů.

Pro vývojáře se zkušeností s knihovnou React nebo samotným jazykem Javascript je snadné s tímto frameworkem začít pracovat. Dále obsahuje také užitečnou funkci Fast refresh, kvůli které není třeba při jakékoliv změně v kódu znovu sestavit celou aplikaci. Změny se při uložení projeví okamžitě v reálném čase, kdy dojde k znovu vykreslení komponent, u nichž došlo ke změně.[6]

Časté aktualizace nejsou příliš konzistentní, pokud se jedná o jejich vydávání, a občas přinášejí problémy u modifikování, neboť změní některé existující postupy a přístupy. Obecně je známo, že React Native není ideální volbou pro rozsáhlejší aplikace, které vyžadují velké množství komplexních animací. To dokládá i příklad jeho samotného autora, Facebook jej totiž využil pouze pro svou obrazovku s událostmi, u nichž se jedná o příklad opakujícího se motivu.



Obrázek 3: Ukázka kódu ve frameworku React Native [7]

4.3.2 NativeScript

Přináší výrazně větší volnost v oblasti psaní kódu, protože stejně jako React Native stojí na javascriptové knihovně, ale na rozdíl od něj si vývojář může zvolit, v jakém z mnohých frameworků chce psát kód. Mezi nejoblíbenější možnosti patří Angular, Vue.js a Typescript. Vezmeme-li

v úvahu tato dvě řešení na podobném základu, tak ze srovnání, v rámci Githubu, vychází až téměř pětkrát populárněji React Native. [8] [9]

Obrovským rozdílem oproti React Nativu, v rámci něhož lze využívat pouze rozšíření a knihovny třetích stran, vytvořených svou rozsáhlejší komunitou, je existence oficiálního obchodu, kde jsou tato rozšíření nabízena. Zde dochází nejen k vydávání nových knihoven, probíhá také aktivní rozšiřování a údržba těch stávajících. Liší se ještě v integraci nativních funkcí, kdy NativeScript nevyžaduje žádnou znalost specifického jazyka platformy, protože jak iOS, tak API od Androidu jsou vkládána do javascriptového virtuálního stroje. [10]

4.4 Multiplatformní kompilátory

Jsou psány ve svém vlastním jazyce (Dart, C#), jeho kód je při spuštění sestaven do nativních kódů dle zvolené platformy.

4.4.1 Flutter

Nejnovějším počinem společnosti Google na poli multiplatformního vývoje byl framework s názvem Flutter. První zmínka se objevila již v roce 2015 v souvislosti s jeho původním jménem SKY, avšak první stabilní verze byla vypuštěna až v prosinci roku 2018. Jeho popularita začala rychle stoupat, což dokládá zájem a využívání frameworku společnostmi, jako je například Alibaba, Tencent, Hamilton či Google Ads. [11]

Aplikace jsou psány v programovacím jazyku Dart, což je objektově orientovaný jazyk postavený na syntaxi jazyku C. Jeho autorem je opět společnost Google. Běží na samostatném jádru a pro uživatelské rozhraní jsou využívány asynchronní metody. Při jeho vývoji se dbalo na jednoduchou a srozumitelnou syntaxi včetně přehledné struktury. Obsahuje také odvození typu proměnné a jak silné, tak slabé typování, což umožňuje snadný přechod z běžných jazyků, jako je Java, C anebo C#. [12]

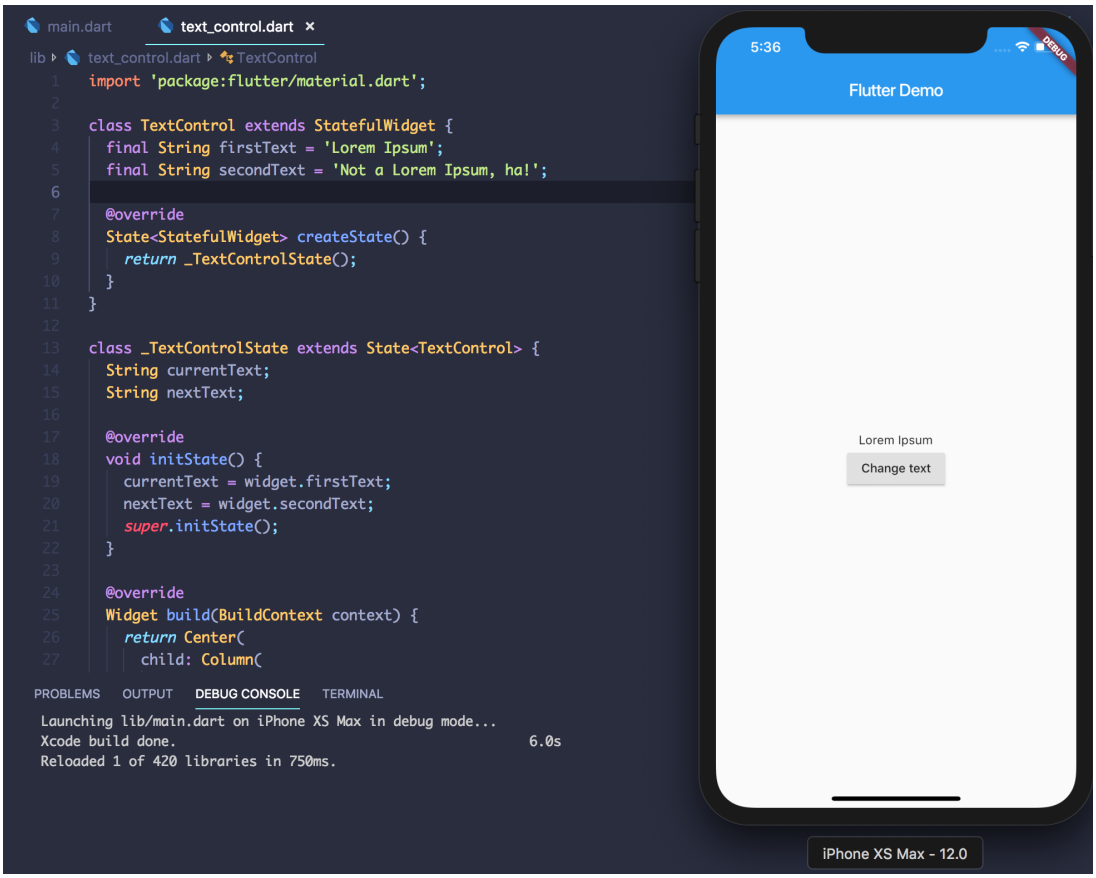
Jazyk Dart je spouštěn na virtuálním stroji Dart VM, jenž poskytuje přístup k nativním API a canvas pro zvolenou platformu. Zde se využívá just-in-time kompilace ve vývojářském módu, přinášející níže zmíněnou funkci Hot reload. Při vydání je v rámci produkčního módu užita ahead-of-time kompilace. [13]

Hlavními stavebními bloky uživatelského rozhraní aplikace vytvořené v rámci Flutteru jsou zde widgety namísto nativních komponent. Všechny prvky, které jsou normálně odděleny, najdeme sjednoceny ve widgetech, ať už se jedná o tlačítko, menu, stylistické vlastnosti, jako je například font, či barvu nebo parametry rozestavění (odsazení, rozestupy). Kromě toho poskytuje navíc i API pro animace, efekty a gesta. Díky této kombinaci není třeba se spoléhat na samotnou platformu a javascriptové propojení, což zajišťuje vyšší výkon.

Jednou z velkých výhod je funkce Hot reload, umožňující snadno a rychle vkládat aktualizované zdrojové kódy do běžícího virtuálního stroje, a framework se postará o opětovné sestavení. Mezi další přednosti Flutteru patří výkonnost srovnatelná s nativně psanými aplikacemi a při vy-

užívání nástrojů od Googlu přijde vhod podobnost s jejich již existujícími aplikacemi, například Google Maps či Gmail.

Několik záporů pro tento stále poměrně nový jazyk vychází právě z jeho stáří. Prozatím neexistuje takové rozsáhlé množství knihoven a existujících řešení jako u jiných zavedených frameworků. Pro některé může být nepříjemností učení se nového jazyka Dart. Dalším problémem je i vyšší velikost souborů, neboť uživatelé mají omezenou paměť na telefonech. Tento problém nicméně postihuje také frameworky React Native a Xamarin.



Obrázek 4: Ukázka kódu ve frameworku Flutter [14]

5 Xamarin

Stejně jako framework Flutter spadá do kategorie multiplatformních kompilátorů, kdy využívá programovací jazyk c#. Obsahuje dva způsoby vývoje multiplatformních aplikací, Xamarin Native a Xamarin.Forms. Pro potřeby naší aplikace se zabýváme výhradně variantou Xamarin.Forms, ve kterém je aplikace napsána.

5.1 Historie

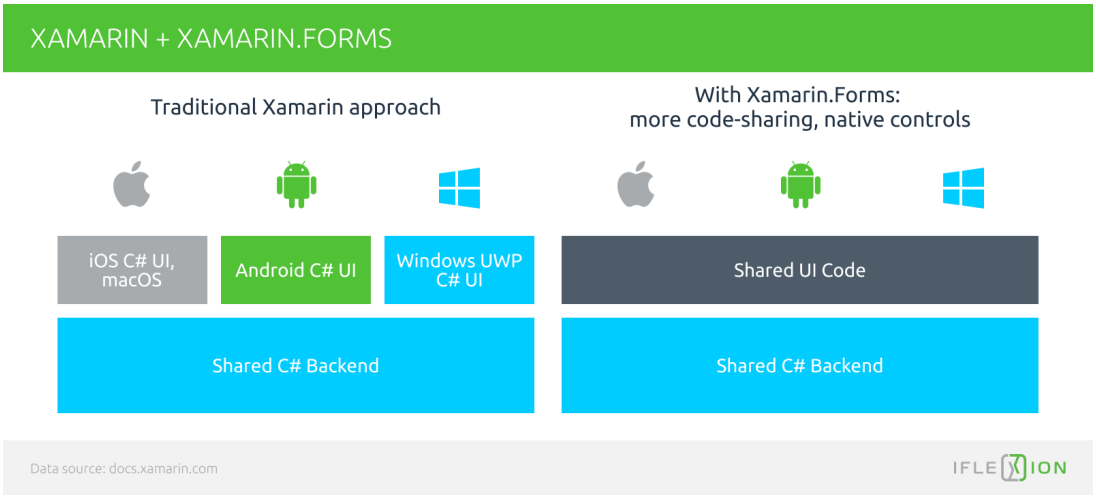
Xamarin vznikl jako projekt Miguela de Icazy a Nata Friedmana, kteří spustili Mono projekt v roce 2001 jako reakci na vydání .NET frameworku společností Microsoft. Hlavním cílem bylo umožnit spouštění tohoto frameworku na jiných platformách, zpočátku hlavně na Linuxu díky akvizici Ximianu. V roce 2009 byl vydán Mono Touch a na iOS App Store se tak objevily první c# aplikace. Ty se musely potýkat s nepříjemnostmi, neboť byly v rámci nových smluvních podmínek od společnosti Apple zakázány. Snažil se totiž udržet kontrolu nad svým vlastním prostředím a odmítal všechny multiplatformní aplikace, nakonec však po vlně kritiky tento zákaz odstranil.

Po akvizici firmy společností Attachmate bylo další pokračování Mono a Mono Touch projektu nejasné. Později však došlo k vytvoření společnosti Xamarin, která dále pokračovala ve stávajícím vývoji. Po dohodě s Attachmate získala navíc práva na Mono, MonoTouch a Mono pro systém Android. Poslední zmíněný byl pak přejmenován na Xamarin.Android. Právě pod touto společností došlo k vydání mnoha úspěšných produktů, v roce 2012 to byl Xamarin.Mac, který otevřel možnost psát v c# aplikace na MacOS. O rok později přichází Xamarin.iOS. Konečně tak dochází k možnosti využívat a sdílet kód mezi operačními systémy iOS, Androidem, MacOS a rovněž Windows Phone.

Microsoft mezitím prošel výraznou změnou jejich směřování a názorů, která vrcholí v roce 2016, kdy získává do svého vlastnictví Xamarin. Vývojářské nástroje Xamarinu jsou od té doby open source a zcela zdarma. Došlo k nahrazení některých komponent těmi od Microsoftu, příkladem může být převod Xamarin Studia na Visual Studio pro Mac. Pokračuje další vývoj a propojování s dalšími akvizicemi od Microsoftu, včetně jejich přímých produktů.

5.2 Xamarin.Forms

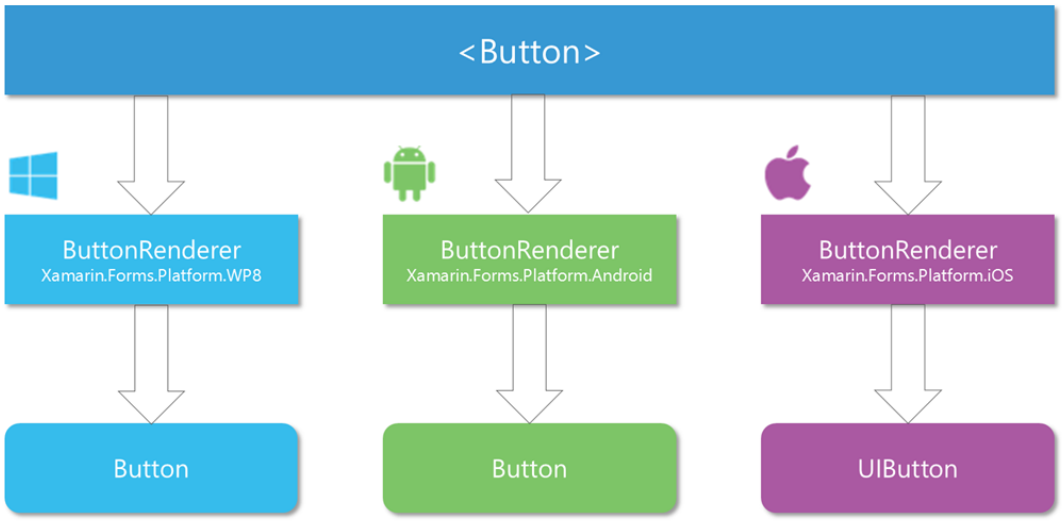
Přichází v roce 2014 jako odpověď na dlouhodobý problém, jenž byl třeba adresovat. Dosud totiž bylo možné psát kód v c# a spouštět jej na různých platformách, avšak uživatelské rozhraní bylo nepřenositelné a nutné při změně cílového systému napsat celé znovu. Přenositelnost tedy stále nebyla optimální a něco takového nestačilo týmu v Xamarinu, který se chtěl dostat co nejblíže hranici sta procent.



Obrázek 5: Srovnání tradičního Xamarinu se Xamarin.Forms
[15]

U obrázku číslo 5 nalevo vidíme klasický Xamarin před příchodem Forms, kdy každý operační systém využíval vlastní uživatelské rozhraní a pouze backendový kód byl sdílený, zatímco u Xamarin.Forms zobrazeném na obrázku napravo je již sdílené i rozhraní.

Této změny se docílilo pomocí přidání abstraktní vrstvy nad uživatelské rozhraní všech platforem, v jejímž rámci probíhá mapování Xamarin prvků na nativní prvky u daného systému. Konkrétní případ lze vidět na obrázku 6. Button (tlačítko) je mapováno na UIButton u iOS a v případě Androidu na Android.Button. To je implementováno za pomoci platformních vykreslovačů, které mají přiděleny všechny ovladače pro každou jednotlivou platformu, vytvářející instanci nativního ovladače. [16]



Obrázek 6: Mapování tlačítka na nativní prvky
[17]

Pro tvorbě uživatelského rozhraní, společného pro obě platformy, jde využít dvou možností. První z nich je pomocí klasického c# a druhou z nich je jazyk XAML, založený na značkovacím jazyku XML. V této práci je, s přihlédnutím lepší přehlednost, používána téměř výhradně druhá možnost, tedy jazyk XAML. Výjimku tvoří pouze případy, kdy je potřeba dynamicky generovat prvky, což lze vidět na obrázku 7.

```
private void FillList(List<EventButton> buttons)
{
    int i = 0;
    while (i < buttons.Count)
    {
        Button button = new Button();
        button.Text = buttons[i].Text;
        button.TextColor = Color.Black;
        button.FontAttributes = FontAttributes.Bold;
        button.FontSize = 16;
        button.HorizontalOptions = LayoutOptions.FillAndExpand;
        button.VerticalOptions = LayoutOptions.FillAndExpand;
        button.BackgroundColor = Color.White;
        if(buttons[i].Popup)
        {
            button.Clicked += Popup_Clicked;
        }
        else
        {
            if(buttons[i].PageName == "MembersPage")
            {
                button.Clicked += Members_Clicked;
            }
            else
            {
                button.Clicked += Button_Clicked;
            }
        }
        Grid.SetRow(button, i);
        Grid.SetColumn(button, 1);
        MyGrid.RowDefinitions.Add(new RowDefinition { Height = new GridLength(40) });
        MyGrid.Children.Add(button);
```

Obrázek 7: Vytváření instancí tlačítek a jejich začlenění do existující mřížky pomocí kódu

6 Návrh aplikace

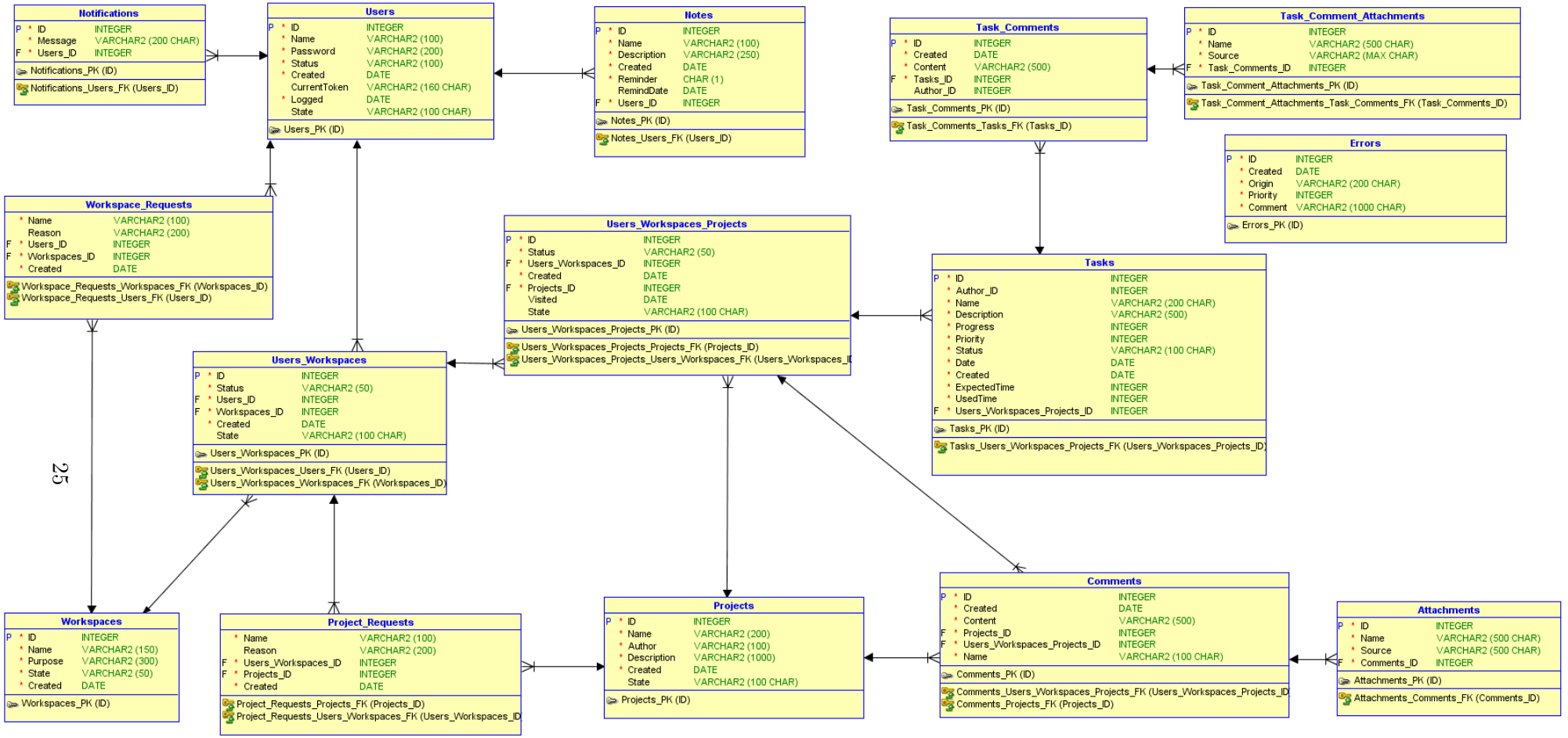
V této kapitole je popsána webová databáze, včetně jejího schématu, kde jsou uložené data aplikace. Z této databáze jsou vybrány a popsány nejdůležitější tabulky. Najdeme zde také rozepsané jednotlivé knihovny, které jsou v aplikaci využívány. Dále je podrobněji rozepsán zvolený systém rolí spolu s Use case diagramem a nakonec ještě tabulky konkrétních případů užití.

6.1 Webová databáze

K aplikaci bylo potřeba vytvořit webovou databázi za účelem uchovávání veškerých dat, se kterými se v aplikaci pracuje. Tato databáze je napsána ve strukturovaném dotazovacím jazyku SQL a umístěna na hostingovém serveru a2hosting.

6.1.1 Schéma databáze

Na obrázku 8 je schéma relačního modelu webové databáze, které bylo vytvořeno pomocí Oracle SQL Developer Data Modeleru.



Obrázek 8: Schéma databáze

Člen pracovní skupiny

Tabulka Users_Workspaces představuje členství uživatele v pracovní skupině a zároveň slouží jako vazební tabulka mezi tabulkou uživatelů a pracovních skupin. Nejdůležitějšími atributy jsou Status a State (stav). První z nich určuje, kdo je uživatel nebo admin, a zda má v dané skupině extra pravomoci popsané v 6.3.2. Stav představuje současnou situaci člena a může nabývat hodnot aktivní (Active) anebo smazaný (Deleted).

Člen projektu

Tabulka Users_Workspaces_Projects představuje členství uživatele v projektu, který je zároveň pod určitou pracovní skupinou. Podobně jako u člena pracovní skupiny zde jsou atributy Status a State. Kromě toho je zde navíc i zaznamenána doba poslední návštěvy projektu, na jejímž základě se pak získávají novinky o dosud nepřečtených komentářích. Ty jsou viditelné v přehledu novinek, viz sekce 7.3.1.

Přílohy

Tabulky Attachments a Task_Comment_Attachments obsahují přílohy ke komentářům v rámci projektů a k jednotlivým úkolům. Jejich součástí je název přílohy, včetně její přípony. Ta je při stažení z databáze využita pro získání příslušného typu internetového média, aby bylo možné uložený soubor otevřít přes nativní nástroje. Dále obsahuje base64 textový řetězec, jenž je vytvořen převodem z pole bytů, přečtených při načtení zvoleného souboru pomocí knihovny Filepicker, viz sekce 6.2.7.

6.2 Využité knihovny

V aplikaci jsou použity následující knihovny, k jejichž stažení posloužil balíčkovací správce NuGet. Jde o Microsoftem podporovaný mechanismus pro sdílení a používání knihoven obsahující hotové řešení. Jako takový nabízí nástroje pro jejich vytváření, ukládání, stažení a zapojení do aplikace. [18]

6.2.1 SQLite-net

Používá se pro ukládání dat v lokální SQLite 3 multiplatformní databázi, ke kterým je potřeba mít rychlý přístup a která jsou častěji opakovaně využívána. Hlavní výhoda této databáze spočívá v jednoduchosti. Například tabulka se v databázi vytvoří podle existující třídy metodou `CreateTable<Typ>()`, jež použije automaticky generované schéma dle zadaného typu. Základní operace pro čtení, zapisování, editaci a mazání jsou rovněž velmi snadné k použití a využívají parametry, čímž je ošetřena bezpečnost vstupů. Při čtení z tabulky lze přes příkaz `[Query<Typ>("select * from Typ")]` používat za pomoci mapování okamžité přetypování pro větší rychlost. [19]

6.2.2 System.Net

Obsahuje dvě třídy, které jsou v aplikaci využívány pro komunikaci s webovou databází přes API. Pro případ, kdy se jedná o GET žádost, jde o třídu `WebClient` a její metodu `DownloadString(URL)` s možností zadat navíc parametry. Druhou třídou je `HttpClient`, kde je užita metoda `SendAsync(request)` pro všechny DELETE a POST žádosti, umožňující zaslat JSON data.

6.2.3 Newtonsoft.Json

Pro komunikaci s databází, umístěnou na webu, je používáno API a veškerá data, která jsou zasílána, mají JSON formát. Newtonsoft.Json nabízí deserializaci a rovněž zpětnou serializaci dat do tříd pro formáty JSON a XML. Pro účely aplikace je využíváno pouze formátu JSON, konkrétně statických metod `SerializeObject(Objekt)` a `DeserializeObject<Class>(JSON)`. Pro účel optimalizace, založené na zasílání pouze nejnutnějších dat, lze využít atribut `[JsonIgnore]`. Podle něj dochází k ignorování vlastností třídy při serializaci do JSONu. Díky své efektivnosti a možnosti užití se jedná v současné době o nejpopulárnější knihovnu balíčkovacího správce NuGet s celkovým počtem téměř 470 milionů stažení. [20]

6.2.4 Microsoft.Azure.NotificationHubs

Knihovna zajištující zasílání push upozornění pro jakoukoliv mobilní platformu. Více lze nalézt v sekci 8.

6.2.5 Rg.Plugins.Popup

Xamarin.Forms umožňuje v základu zobrazit vyskakovací okno s libovolným počtem možností ze seznamu k výběru nebo obsahující textové pole k zadání vstupu. V případě zájmu o pokročilejší formulář, aniž by byla použita navigace a zobrazení nové stránky, zde neexistuje další alternativa. Z toho důvodu je využívána tato knihovna. Nabízí funkci zobrazit libovolnou stránku jako vyskakovací okno spolu s jednoduchou animací. [21]

6.2.6 Xamarin.Essentials

Je-li nutné přistupovat k nativním funkcím telefonu, lze tuto situaci řešit pomocí tzv. Dependency Service. Jeho základem je rozhraní s předdefinovanými funkcemi a k němu vytvořené implementace pro obě platformy. Přitom je nutné jak rozhraní, tak samotné implementace zaregistrovat, aby byly přes Dependency Service nalezeny. Knihovna Xamarin.Essentials obsahuje již implementované rozhraní, které zajišťuje přímou komunikaci s nativními funkcemi ve sdíleném kódu. [22] V aplikaci je využita knihovna PermissionsPlugin, přímo vycházející z Xamarin.Essentials. Její použití je při práci se soubory, konkrétně je využívána pro získání povolení pro práci a ukládání do souborového adresáře. [23]

6.2.7 Xamarin.Plugin.FilePicker

Knihovna umožňující na obou platformách vybrat z adresáře v telefonu libovolný soubor. Takto získaná instance třídy `FileData` obsahuje mimo jiné jméno zvoleného souboru a pole bytů z něj přečtených. To je převedeno do base64 textového řetězce a spolu s názvem souboru uloženo do databáze. [24]

6.3 Systém rolí

V aplikaci tohoto typu je vhodné mít rozdělení pravomocí a práv, podobně jako ve firmách. Toto je zajištěno pomocí systému rolí, který je odstupňovaný a začíná od role s nejvyšším počtem pravomocí. Hlavním rozdílem mezi vedoucími projektu nebo pracovní skupiny a adminem je v rámci, ve kterém platí jejich pravomoci. Ten se postupně snižuje od celé aplikace přes pracovní skupiny až po vedoucí jednotlivých projektů.

6.3.1 Admin

Jedná se o správce aplikace, který má nejvyšší možnou roli. Ve všech pracovních skupinách a projektech má automaticky stejná práva jako jejich vedoucí a může do nich vstupovat, i když není jejich členem.

6.3.2 Vedoucí pracovní skupiny

Je vybrán vždy zakladatelem pracovní skupiny, jež má tuto pozici přiřazenou automaticky jakožto její tvůrce. V jejím rámci může přidávat a přijímat do pracovní skupiny další lidi, nebo je z ní naopak odebírat. Dále má právo smazat libovolný projekt z pracovní skupiny a v rámci nich má stejné pravomoci jako jejich vedoucí. Má na starosti plynulé fungování pracovní skupiny a spolupráci s vedoucími projektu.

6.3.3 Vedoucí projektu

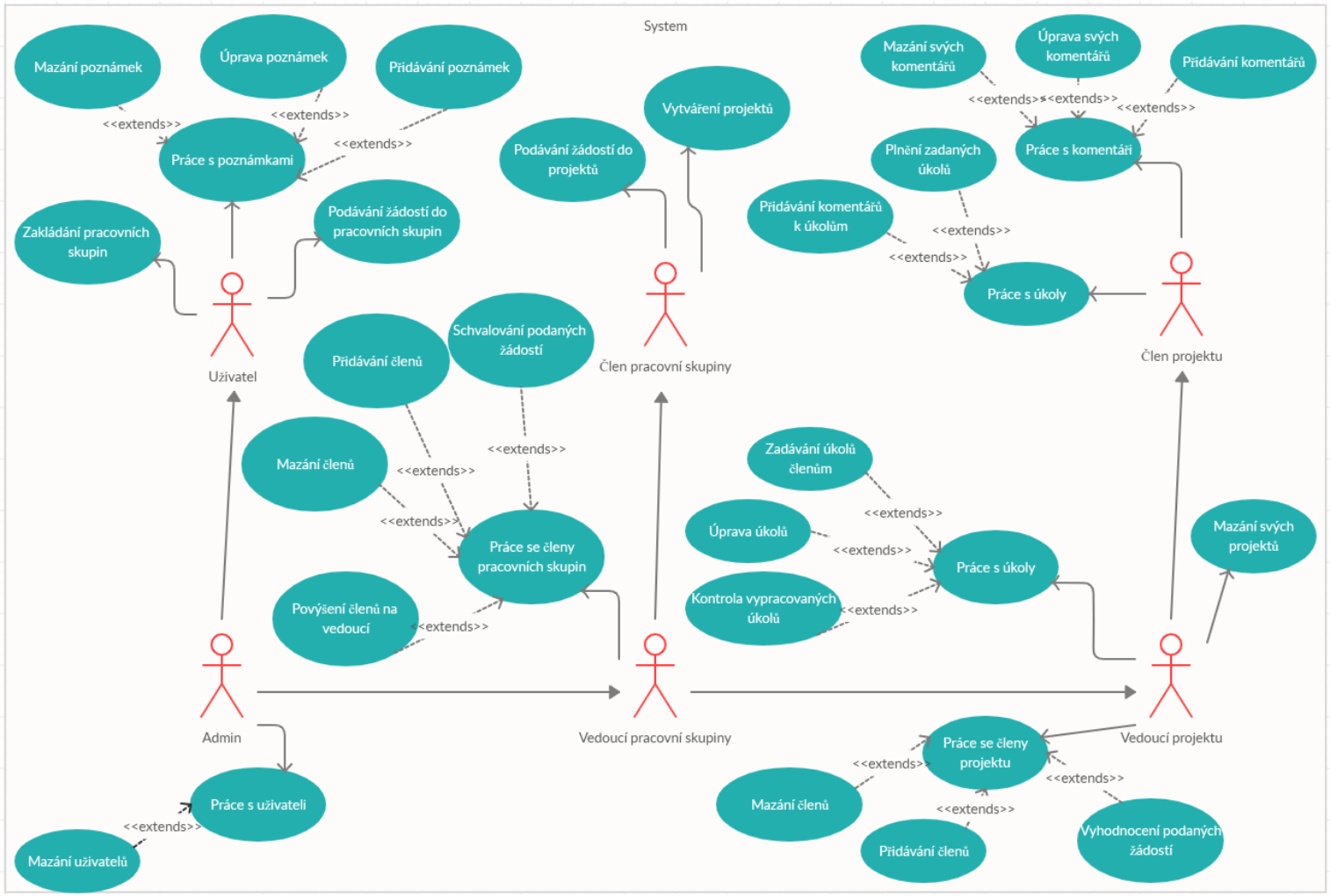
Podobně jako vedoucí workspace je vždy vybrán zakladatelem projektu, který je opět jeho vedoucím. V jeho rámci má stejné pravomoci, to jest přidávat a přijímat do projektu další lidi, či je z něj naopak odebírat, zadávat úkoly, hodnotit jejich plnění. Dále může mazat a upravovat všechny existující komentáře. Jeho hlavním úkolem je zajistit zdárné plnění přidělených úkolů a dohlížet na dokončení projektu jako celku.

6.3.4 Člen projektu

Je to běžný uživatel nemající žádná práva navíc. Buď je začleněn vedoucím, nebo si o členství může sám zažádat. Stará se o plnění zadaných úkolů, přidává, maže a upravuje vlastní příspěvky.

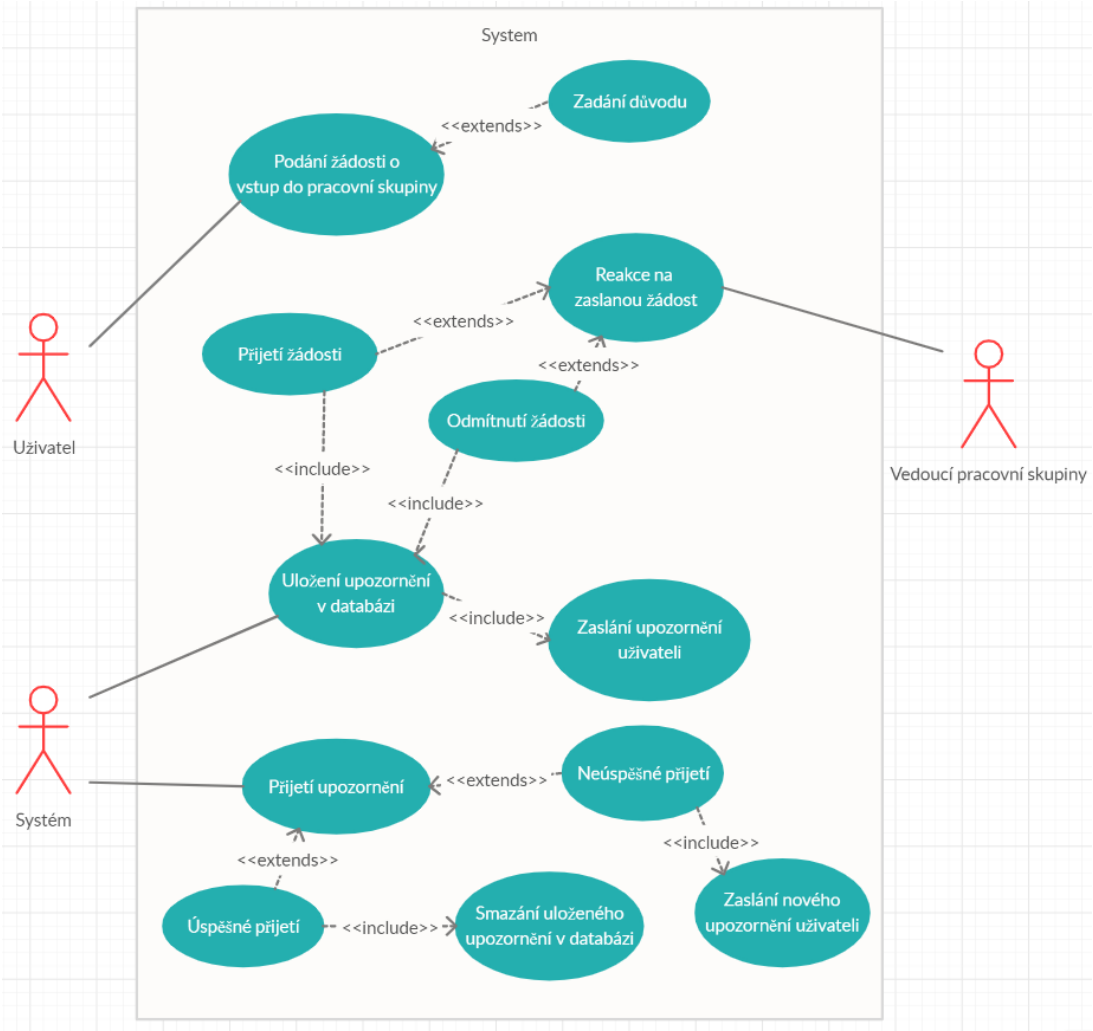
6.3.5 Use Case Diagram

Zde jsou detailněji vymezené výše definované role a funkce, jež jsou součástí aplikace, a ke kterým mají aktéři přístup pomocí Use case diagramu.



Obrázek 9: Use case uživatelských rolí

6.3.6 Specifické případy užití



Obrázek 10: Use Case diagram žádosti o vstup do pracovní skupiny a její vyhodnocení vedoucím

Název případu užití	Žádost o vstup do pracovní skupiny a její vyhodnocení vedoucím		
Identifikace případu užití	UC1		
Cíl případu užití	Úspěšné přijetí do skupiny		
Primární aktéři	Uživatel, Vedoucí pracovní skupiny		
Pomocní aktéři			
Vstupní podmínky	Uživatel si vybere pracovní skupinu, kde není členem, a podá žádost.		
Výstupní podmínky	Žádost uživatele je přijata.		
Základní scénář	**Krok**	**Role**	**Akce**
	1	Systém	Systém vypíše existující pracovní skupiny.
	2	Aktér	Uživatel zvolí pracovní skupinu, kde není členem.
	3	Systém	Systém vyzve uživatele k zadání důvodu pro vstup.
	4	Aktér	Uživatel zadá důvod a odešle žádost.
	5	Systém	Systém uloží žádost do databáze.
	6	Aktér	Vedoucí si vyžádá zobrazení existujících žádostí.
	7	Systém	Systém vypíše vedoucímu žádost s uvedeným důvodem.
	8	Aktér	Vedoucí schválí žádost.
	9	Systém	Systém vytvoří uživateli členství v pracovní skupině.
	10	Systém	Systém zašle upozornění na přijatou žádost uživateli.
	11	Systém	Systém vytvoří žádost v databázi.
	12	Systém	Systém úspěšně přijme upozornění pro daného uživatele.
	13	Systém	Systém smaže upozornění v databázi.
Alternativní scénář	12a	Systém	Nedojde k úspěšnému přijetí upozornění.
	12b	Aktér	Aktér se přihlásí na jiném zařízení.
	12c	Systém	Systém znovu odešle nové upozornění na přijetí žádosti.
	12d	Systém	Následuje krok 12.

Obrázek 11: Use Case diagram zadání a vyřešení úkolu

Název případu užití	Zadání a vyřešení úkolu		
Identifikace případu užití	UC2		
Cíl případu užití	Vyřešení úkolu		
Primární aktéři	Zadavatel, Člen projektu		
Pomocní aktéři			
Vstupní podmínky	Zadavatel je vedoucím stejného projektu jako zvolený člen.		
Výstupní podmínky	Úkol je označen jako vyřešený.		
Základní scénář	**Krok**	**Role**	**Akce**
	1	Aktér	Zadavatel si ve všeobecném přehledu zvolí přidání úkolu.
	2	Systém	Systém vypíše projekty, kde je členem či vedoucím.
	3	Aktér	Zadavatel zvolí projekt, kde je vedoucím.
	4	Systém	Systém vybere všechny členy a zobrazí je k výběru.
	5	Aktér	Zadavatel vybere člena, kterému bude úkol přidělen.
	6	Aktér	Zadavatel vyplní úkol a přidá jej.
	7	Systém	Systém uloží úkol do databáze.
	8	Aktér	Uživatel si vyžádá seznam úkolů.
	9	Systém	Systém vypíše seznam úkolů.
	10	Aktér	Uživatel splní úkol dle zadání.
	11	Aktér	Uživatel úkol označí za vyřešený.
	12	Systém	Systém úkol aktualizuje a uzavře.
Alternativní scénář	1a	Aktér	Zadavatel si zvolí přidání úkolu u konkrétního člena projektu.
	1b	Aktér	Následuje krok 6.
	10a	Aktér	Uživatel přidá komentář s řešením.
	10b	Aktér	Zadavatel zkontroluje řešení a označí úkol za vyřešený.
	10c	Systém	Následuje krok 12.

7 Implementace

V této části je popsána implementace hlavních funkcí vytvořené aplikace. Dále jsou uvedeny příklady uživatelského rozhraní, například všeobecný přehled či stránka pracovní skupiny. Konečná část je věnována popisu lokální databáze a jednotlivých dat, která jsou takto ukládána, včetně příkladů využití.

7.1 Poznámky

Slouží v situaci, kdy si uživatel potřebuje zapsat své myšlenky k některému z projektů, k pracovním skupinám, případně samostatně. Kromě případů obyčejných poznámek mohou navíc nabývat i charakteru připomínek, což se pozná pomocí vlastnosti Reminder. Při zvolení této možnosti se jednoduše nastaví datum a čas a na jejich základě dojde k zaregistrování nativního alarmu s názvem a popisem připomínky. S poznámkami je možné pracovat pouze ve všeobecném přehledu, kde je lze editovat, mazat či přidávat.

7.2 Úkoly

V systému pro správu projektů jsou úkoly velmi důležitou součástí. Samotný úkol obsahuje ID jeho zadavatele, název, popis, momentální postup v jeho plnění (stupňovaný po 25 od 0 do 100) a závažnost čili prioritu. Dále současný stav, v jakém se úkol nachází. Možné stavy jsou nový, řešený, vyskytl se problém a dokončený. Součástí je také datum, dokdy by měl být úkol vyřešen, jak dlouhou dobu by měl trvat, a počet hodin skutečně strávených jeho plněním. Úkoly jsou vždy zadávány v rámci projektu jeho členům.

Jsou dva způsoby zadávání úkolů. První z nich je prováděný ve všeobecném přehledu, viz sekce 7.3.3. Zde dojde, na základě uživatelského ID, ke shromáždění všech projektů, jejichž je členem, včetně informace, zdali je vedoucím. Úkoly totiž v projektu může všem členům zadávat pouze vedoucí projektu, aby byla zachována struktura rolí. V případě, že je uživatel pouhým členem, lze zadat úkol pouze sám sobě. Druhou možností je využít seznam členů projektu, viz sekce 7.3.8, kde se přidělují úkoly přímo konkrétním uživatelům.

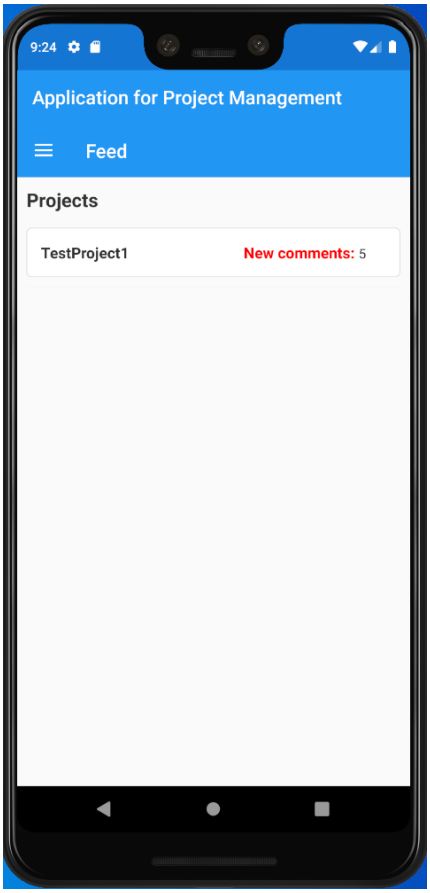
Ve všeobecném přehledu si lze zobrazit úkoly zadané nebo ty, které je potřeba splnit. Existuje zde možnost řadit je podle termínu dokončení, důležitosti, míry pokroku nebo pohromadě s ostatními dle projektu. V kombinaci s tímto řazením lze vybírat úkoly ještě dle jejich současného stavu, což zajišťuje snadný přehled.

7.3 Uživatelské rozhraní

Následující podkapitola se věnuje popisu zobrazených obrázků uživatelského rozhraní. Z důvodu velkého množství stránek a vyskakovacích oken v aplikaci jsou popsány pouze nejzajímavější z nich.

7.3.1 Novinky

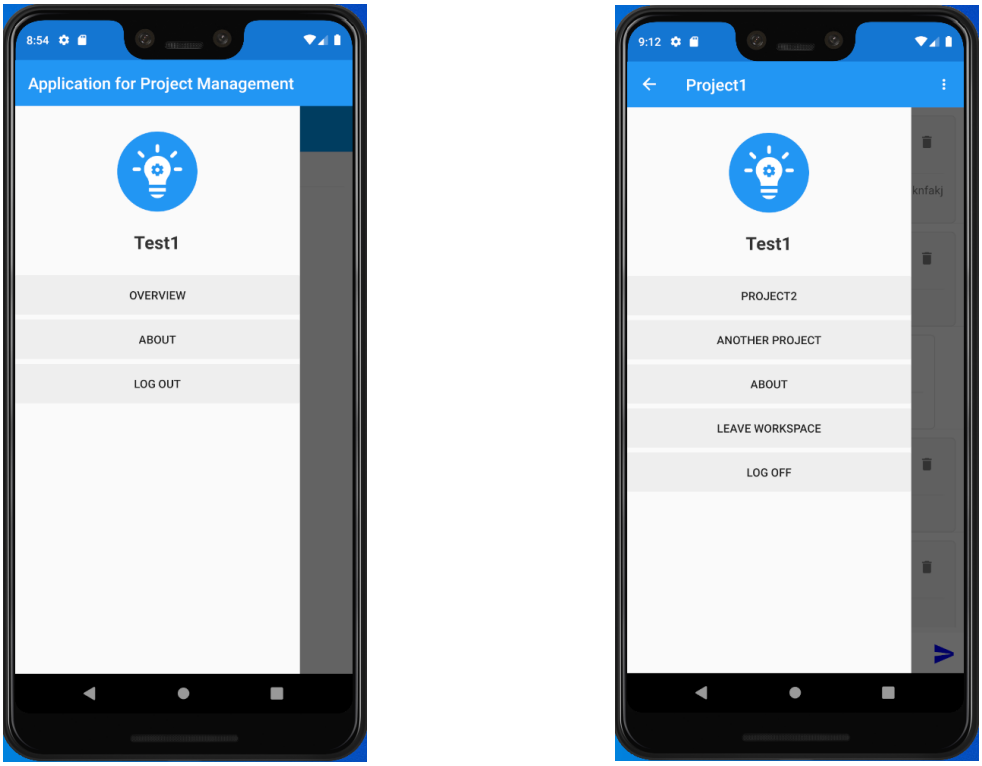
Jakmile dojde k úspěšnému přihlášení, uživateli je zobrazena stránka s novinkami, viz obrázek 12. Ta obsahuje přehled o uskutečněných akcích v systému. Konkrétně jsou zde vyobrazeny nové komentáře v projektech dle poslední návštěvy a nově zadané úkoly .



Obrázek 12: Úvodní stránka s přehledem proběhlých událostí

7.3.2 Boční menu

Pro orientaci a navigaci v aplikaci je využíváno boční menu. Jeho obsah se dynamicky mění v závislosti na současné poloze v aplikaci. Vždy jsou dostupná tlačítka About (informace o aplikaci) a Log out. Po stisknutí tlačítka Log out dojde k přechodu do přihlašovací stránky a vymazání dat z lokální databáze. Na obrázku 13a lze vidět boční menu u stránky novinek. Při vstupu do pracovní skupiny se přidává navíc tlačítko pro opuštění skupiny (Leave Workspace). Speciálním případem je boční menu u některého z projektů, viz obrázek 13b. Zde jsou navíc přidány ostatní projekty ve skupině, v nichž je uživatel členem. Toto slouží pro rychlý přesun mezi zobrazenými projekty.

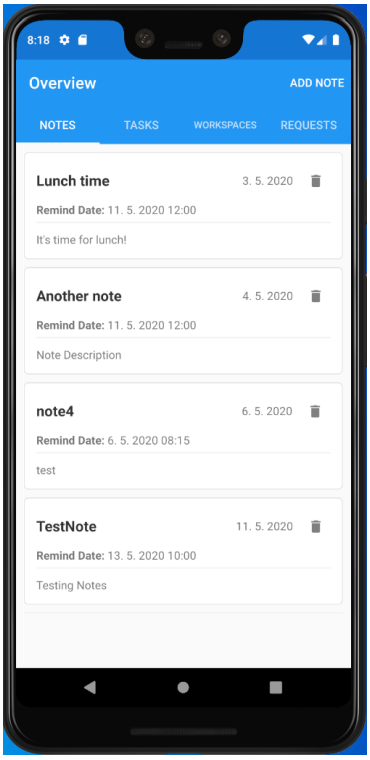


(a) Boční menu u stránky s novinkami (b) Boční menu při vstupu do stránky projektu

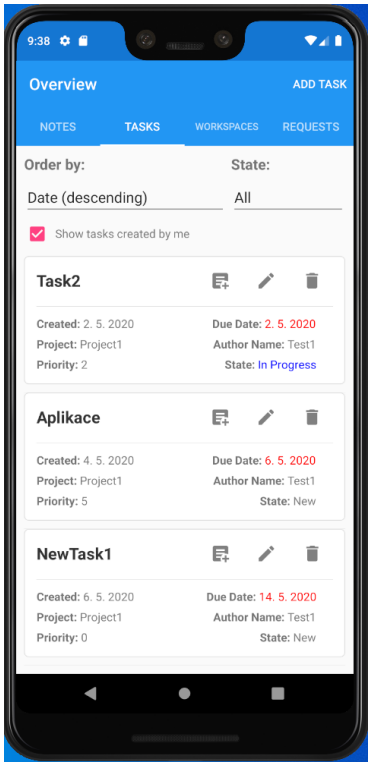
Obrázek 13: Ukázka bočního menu

7.3.3 Celkový přehled

Celkový přehled (Overview) slouží pro všeobecnou orientaci uživatele. Důležitý je zde seznam karet, v němž najdeme celkem čtyři záložky. První z nich, viditelný přímo na obrázku 14a, zobrazuje seznam poznámek uživatele, se kterými lze s nimi pracovat pouze v tomto přehledu. Další karty pak obsahují záložku se všemi úkoly uživatele, viz obrázek 14b, kartu pro práci s pracovními skupinami a konečně přehled o všech zaslaných či přijatých žádostech o vstup do pracovní skupiny.



(a) Celkový přehled aplikace s listem poznámek

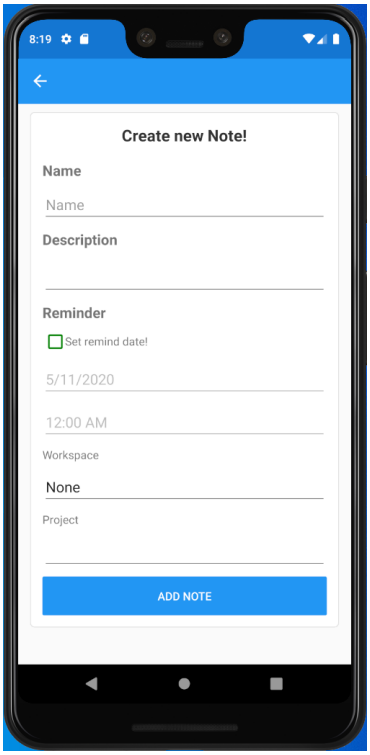


(b) Celkový přehled aplikace s listem úkolů

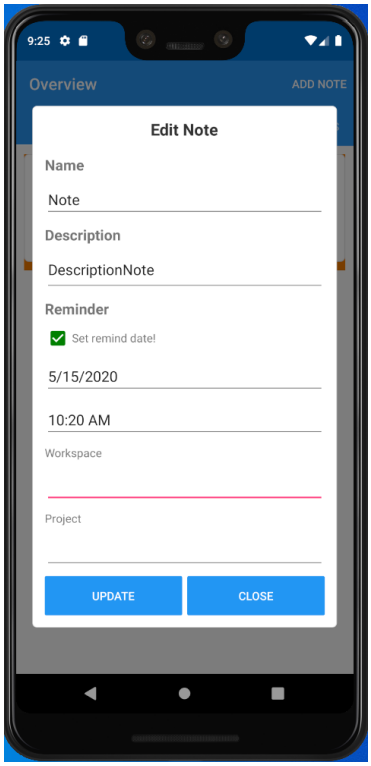
Obrázek 14: Záložky celkového přehledu

7.3.4 Poznámky

Po stisknutí tlačítka Add note v přehledu lze přidávat nové poznámky pomocí formuláře viditelného na obrázku 15a. Při kliknutí na některou z existujících poznámek dochází k zobrazení vyskakovacího okna umožňující její úpravu, viz obrázek 15b.



(a) Přidání nové poznámky

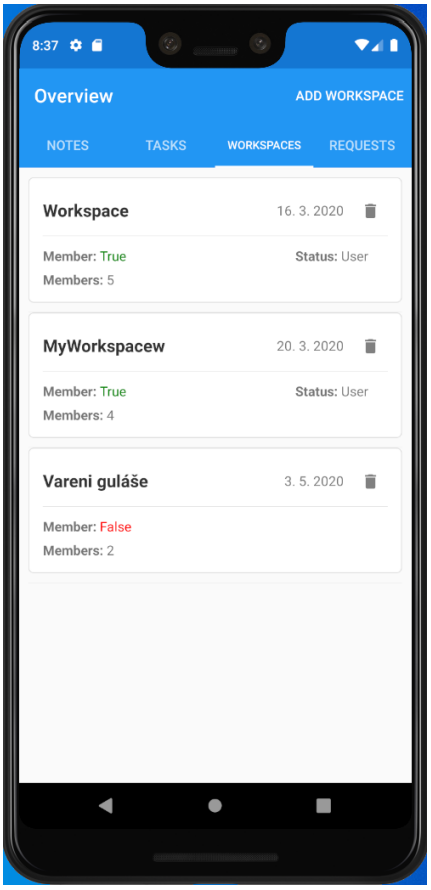


(b) Úprava stávající poznámky

Obrázek 15: Práce s poznámkami

7.3.5 Pracovní skupiny

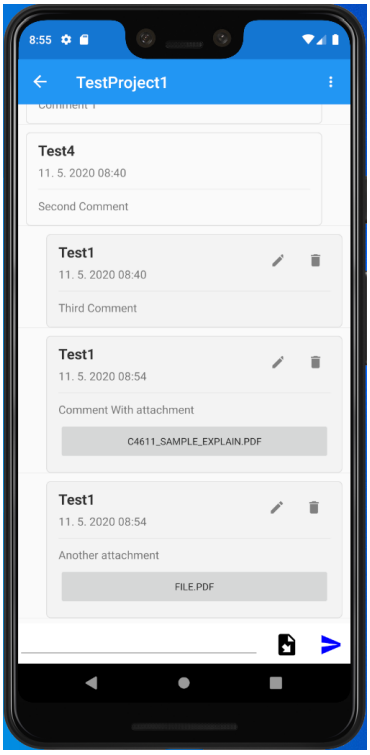
U jednotlivých pracovních skupin jsou viditelné informace o datu založení, zdali je současný uživatel členem této skupiny, a pokud ano, jaký v ní má status. Dostupná je ještě informace o počtu současných členů, a jestliže je adminem této skupiny, pak i tlačítka pro smazání a úpravu. Při kliknutí na libovolnou pracovní skupinu dojde na základě informace o členství ke dvěma možným scénářům. První z nich počítá s aktivním členstvím a nabídne uživateli vstup do jejího přehledu, viz obrázek . Při druhém vstupu dochází k nabídce textového pole jako na obrázku , kde je zadán důvod pro vstup a odeslána žádost.



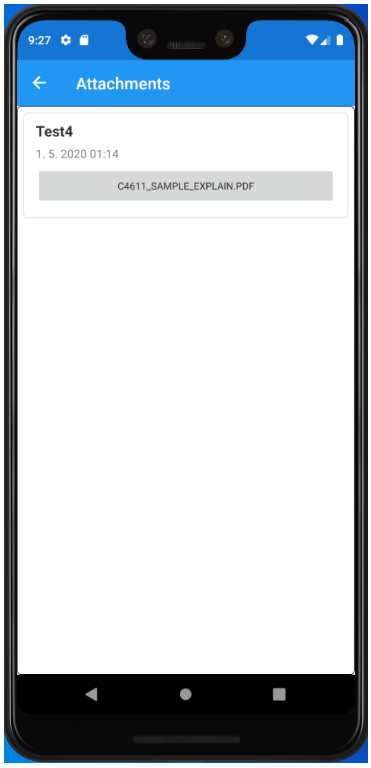
Obrázek 16: Celkový přehled aplikace s listem pracovních skupin

7.3.6 Projekt

Na obrázku 17a je viditelná stránka s komentáři u projektu. Vlastní komentáře jsou jednoznačně rozlišeny rozdílnou barvou, umístěním a přítomností tlačítek umožňujících komentář buď smazat, nebo jej upravit. Komentáře mohou obsahovat také přílohu, při kliknutí na ni dojde ke stažení souboru. Všechny přílohy v rámci projektu si lze přehledně zobrazit na stránce Attachments, viz obrázek 17b.



(a) Stránka s komentáři k vybranému projektu

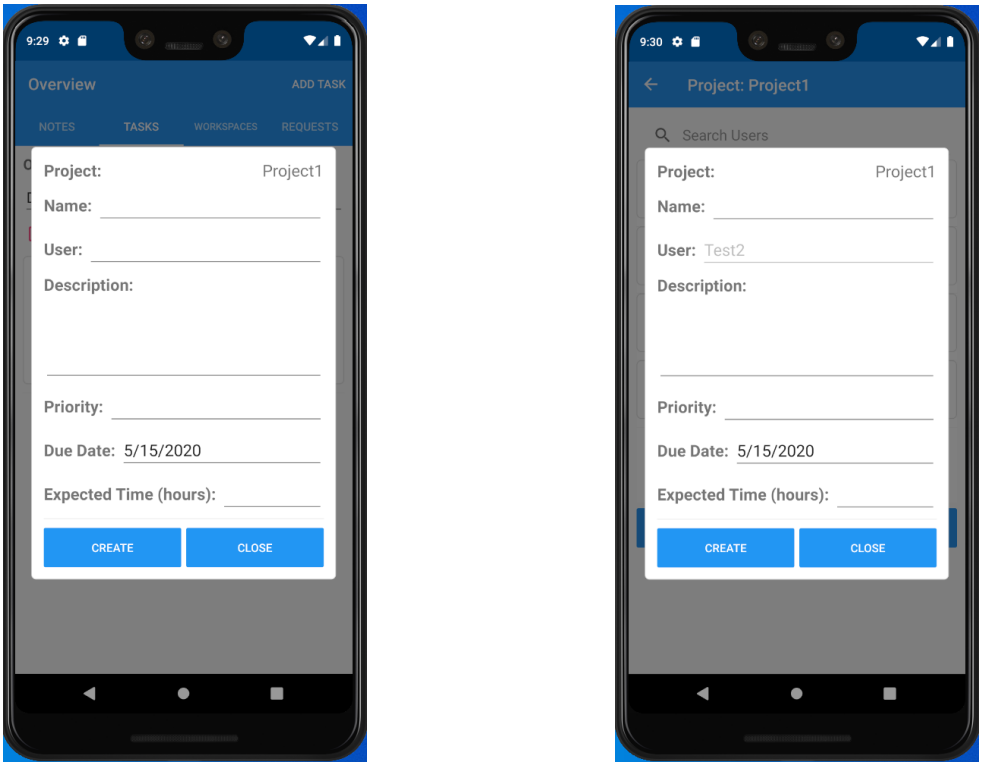


(b) Stránka s přílohami k vybranému projektu

Obrázek 17: Stránky projektu

7.3.7 Zadávání úkolů

Jak již bylo popsáno v sekci 7.2, úkoly lze zadávat dvěma způsoby. První je přes záložku všeobecného přehledu, kde jako první dojde ke zvolení projektu, v jehož rámci se bude přidělovat úkol, tento můžeme spatřit na obrázku 18a. Druhá varianta jde cestou přes vypsané uživatele projektu, viz obrázek 19a. Při této možnosti je již napevno vybrán člen projektu a nelze jej při zadávání měnit, jak lze vidět na obrázku 18b.

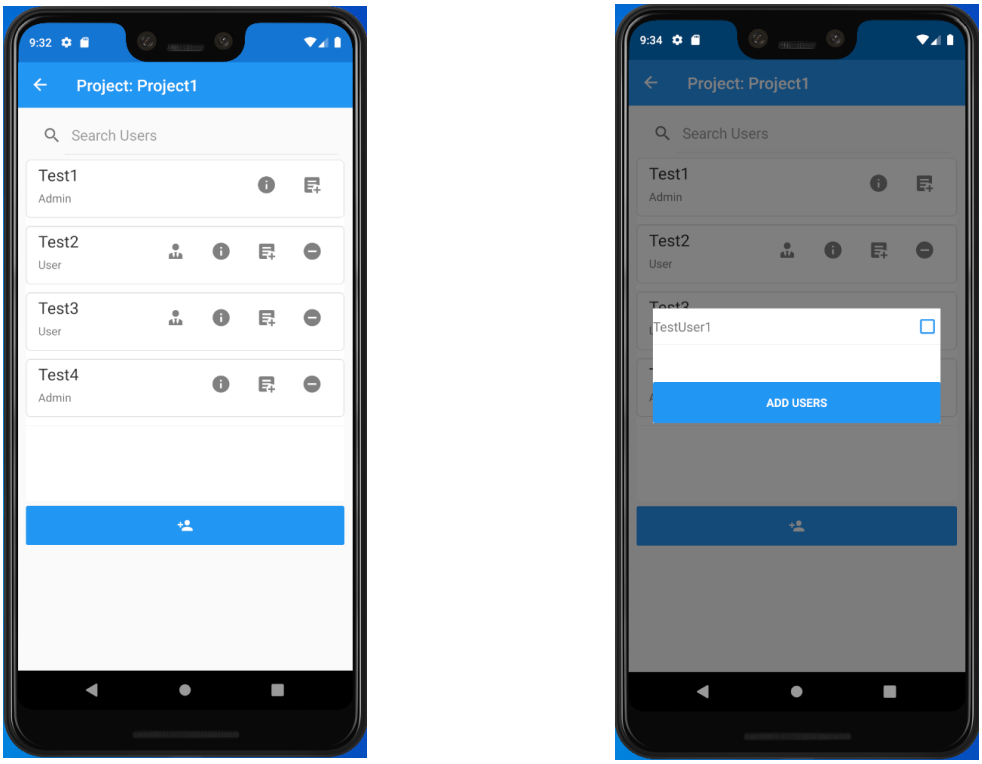


(a) Zadání úkolu přes všeobecný přehled (b) Zadání úkolu přímo uživateli v rámci projektu

Obrázek 18: Zadávání úkolu

7.3.8 Seznam členů

Obrázek 19a obsahuje vypsaný seznam členů projektu, u kterých je možné přejít na jeho detail. Pokud je vstupující uživatel adminem daného projektu, pak má ještě odemknuté možnosti - může přidat extra práva, zadat úkoly anebo člena odstranit. V tomto případě je umožněno i přidávání uživatelů viditelné na obrázku 19b.



(a) Seznam členů vybraného projektu

(b) Přidávání nových členů do projektu

Obrázek 19: Práce se členy projektu

7.4 Lokální databáze

Z důvodů optimalizace a interaktivity jsou některá stažená data ukládána lokálně přímo v telefonu, aby k nim bylo možné téměř okamžitě přistupovat. Kromě toho je pro uživatele výhodou, že i při zavření aplikace zůstává zaznamenán její stav. V případě opětovného zapnutí tak dojde k přechodu do původního bodu. Pro tento účel je využívána SQLite-net knihovna, popsaná v sekci 6.2.1. Dále jsou uvedena jednotlivá data, která aplikace uchovává, a je zmíněno, v jakých případech k takové akci dochází. Při zvolení možnosti odhlášení na bočním menu, viz sekce 7.3.2, dochází k vymazání všech uložených dat z databáze.

Uživatel

Uživatel je jádrem systému, na něhož je vše postupně navázáno. Může se jednat o poznámky, úkoly, upozornění nebo členství v pracovních skupinách a projektech. K uložení současného uživatele do lokální databáze dochází vždy při úspěšně vykonaném přihlášení. Ten je pak využit při získávání nejnovějších událostí, které v aplikaci proběhly od jeho posledního přihlášení, viz sekce 7.3.1. Dále je používán při vstupu do všeobecného přehledu, kde má uživatel pohromadě všechny poznámky, úkoly, žádosti a pracovní skupiny. Poslední případy užití jsou u vytváření vlastních poznámek, nových pracovních skupin či žádostí o vstup do nich.

Pracovní skupina

Obsahuje její název, důvod existence, současný stav a datum, kdy byla založena. K jejímu uložení dochází spolu se členem pracovní skupiny, jejíž je součástí. Slouží pro získávání projektů a členů existujících v uložené pracovní skupině.

Člen pracovní skupiny

K jeho uložení dochází vždy při vstupu do zvolené pracovní skupiny, jejímž je členem. Slouží pro získávání informace o členství v projektech, které existují v pracovní skupině, a k rozlišení jeho práv, viz sekce 6.1.1.

Projekt

Obsahuje název, jméno zakladatele, popsaný důvod existence a datum založení projektu. K jeho uložení dochází spolu se členem projektu, jehož je součástí. Slouží pro získávání komentářů, včetně příloh a členů existujících v uloženém projektu.

Člen projektu

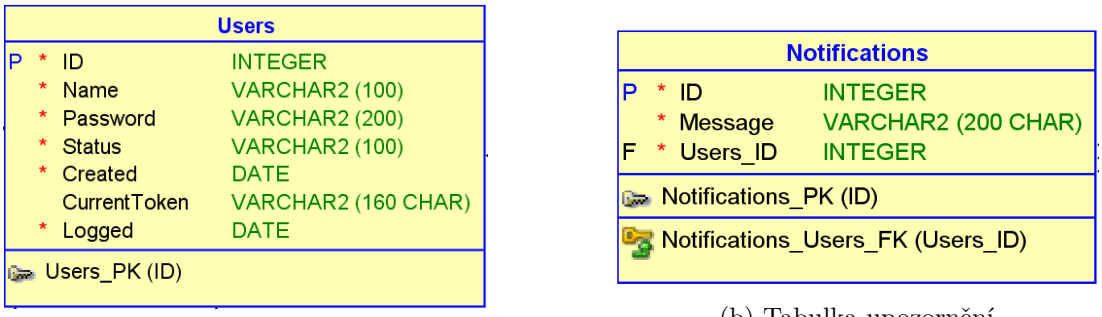
K jeho uložení dochází vždy při vstupu do zvoleného projektu, jehož je členem. Slouží pro identifikování vlastních komentářů, které existují v pracovní skupině, a k rozlišení jeho práv, viz sekce 6.1.1.

8 Upozornění

Důležitou součástí aplikace jsou upozornění na různé akce v systému, konkrétně oznamující výsledek vyhodnocení uživatelovy žádosti o vstup do pracovní skupiny či projektu. S upozorněními není možné pracovat přímo přes sdílený kód, ale je třeba implementovat jejich řešení pro každou platformu zvlášť. Hlavní část práce s upozorněními je postavena na modulu Azure Notification Hubs, přes který jsou upozornění zasílána a který zajišťuje komunikaci s platformním řešením Androidu.

8.1 Princip

Každý uživatel má v databázi uložený svůj současný kód zařízení (CurrentToken), viz obrázek 20a, který je vždy při přihlášení obměněn, pokud dojde k jeho změně. Při vyhodnocení žádosti je odesláno upozornění s využitím uloženého kódu zařízení. Existuje však ještě možnost, že se uživatel od doby zaslání odhlásí a vymění stávající telefon za jiný. Pro takové případy existuje v databázi tabulka Notifications, viz obrázek 20b, kde jsou odeslaná upozornění zapsána a k jejich vymazání dojde až po úspěšném přijetí odpovídajícím uživatelem. Ve zmíněném případě změny zařízení dojde po novém přihlášení a zjištění nedoručených upozornění k jejich opětovnému zaslání. Celý tento případ užití je podrobněji popsán v UC1, nalézající se v sekci 6.3.6.



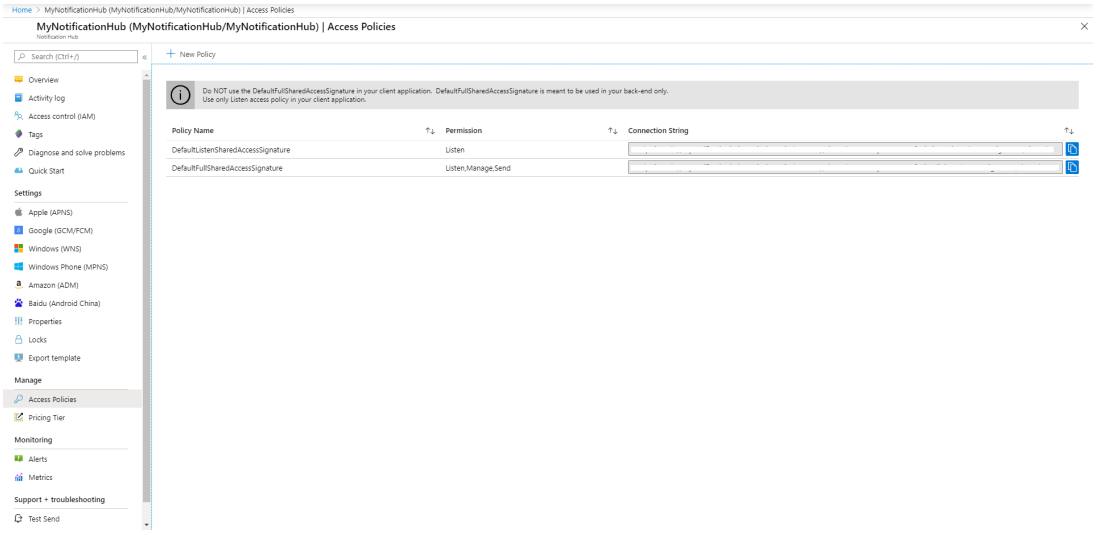
(a) Tabulka uživatelů

(b) Tabulka upozornění

Obrázek 20: Tabulky uživatelů a upozornění ve webové databázi

8.2 Azure Notification Hubs

Modul umožňující zasílat oznámení do zařízení se systémy Android, iOS, Windows nebo Kindle pomocí platformních služeb jako FCM (Firebase Cloud Messaging), APNs (Apple Push Notification service) a dalších. Pro potřeby této aplikace je využívána pouze zmíněná služba FCM. Důležitou součástí je tzv. Access Policies, viditelná na obrázku 21. Zde se nachází textové řetězce obsahující parametry potřebné pro příjem a zasílání upozornění. Velkou výhodou je vysoký potenciál škálovatelnosti, kdy je možné zaregistrovat miliony zařízení a rozeslat miliardy upozornění. [25]

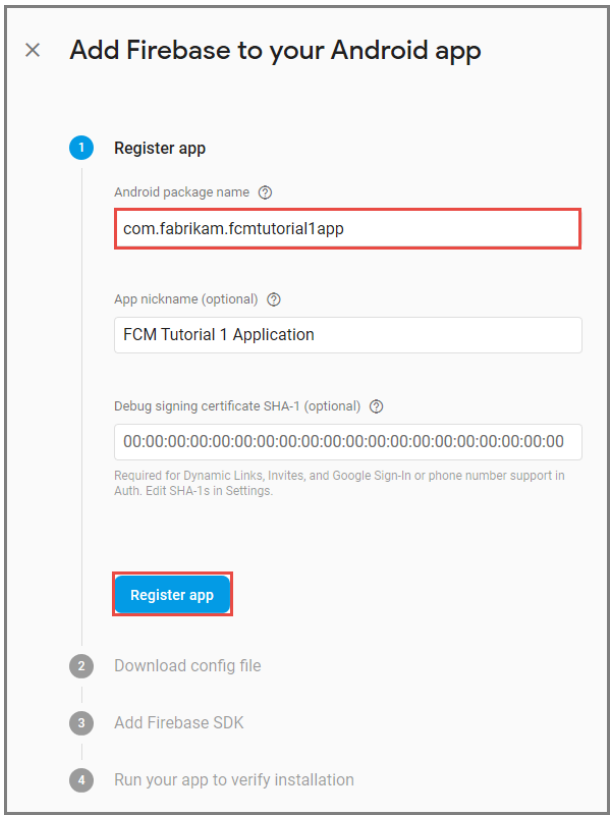


Obrázek 21: Access policies Azure Notification Hub

8.3 Firebase Cloud Messaging

Pro úspěšnou integraci na operačním systému Android je využívána služba Firebase Cloud Messaging. Zde je nutné mít zaregistrovaný účet a vytvořený projekt. Poté dojde k zadání názvu Android balíku, zaregistrování aplikace dle obrázku 22, stažení vygenerovaného JSON souboru a

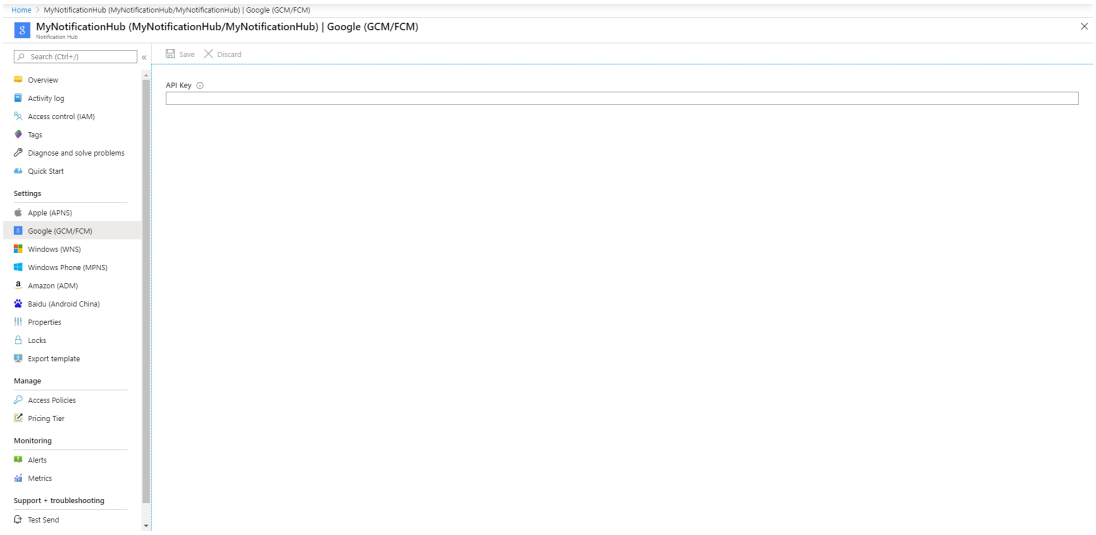
jeho vložení do složky Android projektu. Následuje stažení knihovny Xamarin.Firebase.Messaging přes balíčkovacího správce NuGet. [26]



Obrázek 22: Registrace aplikace do služby Firebase [26]

V souboru AndroidManifest.xml se přidají potřebná povolení a přijímače. V Android projektu je dále vytvořena třída FirebaseService, která dědí z třídy FirebaseMessagingService, zajišťující získání nového kódu zařízení a jeho uložení do databáze. Kromě toho implementuje registraci samotného zařízení do ANH a přijetí zaslaných upozornění. Ve třídě MainActivity dochází k vytvoření komunikačního kanálu s využitím dat poskytnutých již vytvořenou třídou FirebaseService.

Jakmile jsou všechny předchozí kroky splněny, dochází k přidání vygenerovaného API klíče z FCM do ANH, což je vidět na obrázku 23, a tím je integrace pro tuto platformu završena.

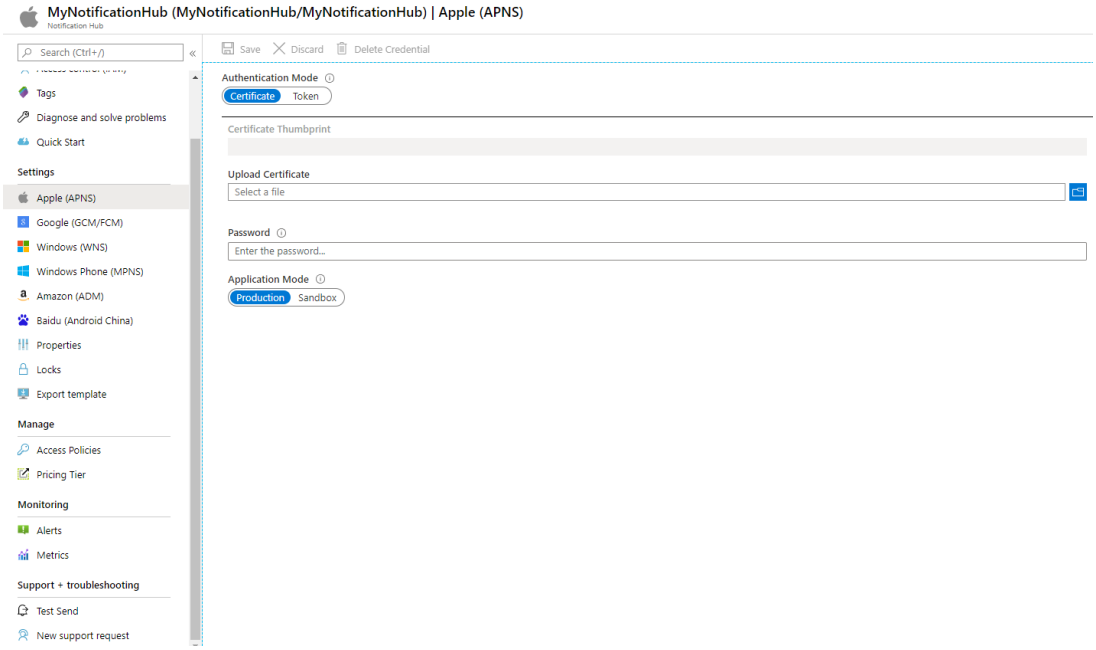


Obrázek 23: Připojení Firebase Cloud Messaging do Azure Notification Hubs

8.4 Apple Push Notification service

Pro upozornění na operačním systému iOS se využívá APNs. Pro jeho úspěšnou implementaci je třeba dodržet následující postup. Mezi nutné předpoklady patří připravené ANH, vlastnictví Apple zařízení a zaplacení iOS vývojářského účtu. Jedná se pouze o cestu integrace za využití certifikátu, možnost přidání pomocí kódu zde není uvedena. [27]

1. Vygenerování a uložení certifikátu pomocí programu Keychain Access na Apple zařízení.
2. Zaregistrování mobilní aplikace ve vývojářském účtu, zde je potřeba ještě zvolit, aby obsahovala funkcionalitu upozornění.
3. Nahrání vygenerovaného certifikátu do vývojářského účtu.
4. Vygenerování a stažení SSL certifikátu ve vývojářském účtu.
5. Vytvoření zřizovacího profilu pro aplikaci.
6. Uložení staženého certifikátu do ANH, viz obrázek 24.
7. V iOS projektu aplikace, konkrétně v Info.plist, je potřeba ověřit, že identifikace sady odpovídá té zadané do vývojářského účtu.
8. V Entitlements.plist je třeba povolit push upozornění.
9. Stažení a instalace knihovny Xamarin.Azure.NotificationHubs.iOS pomocí NuGet.
10. Přidání tříd obsahující metody pro vytvoření komunikačního kanálu a přijímání upozornění.



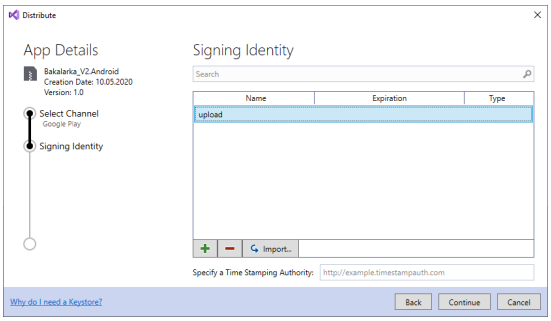
Obrázek 24: Připojení Apple Push Notification service do Azure Notification Hubs

9 Google Play

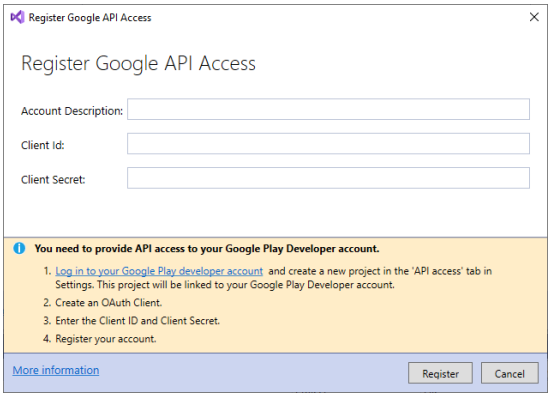
O jeho oficiální existenci lze hovořit od roku 2012, kdy se původní Android Market přejmenoval na nynější Google Play. Jedná se o oficiální obchod s aplikacemi pro systém Android. Nabízí také hudbu, online knihy, filmy a televizní programy. Se svým dosahem, zahrnujícím více než 190 zemí a teritorií po celém světě a více než jednou miliardou aktivních uživatelů, jde o největší distribuční prostor pro vyvinuté aplikace. Kromě vydání veřejné aplikace do obchodu je možné využít testování nahrané aplikace pro pozvané uživatele. Zde je třeba rozlišovat mezi dvěma typy vydávacích prostředí. Prvním z nich je Alfa verze, využívaná pro distribuci aplikace menšímu počtu uživatelů bez nutnosti získání plné revize ze strany Google, a je tedy vhodná pro interní testování. Beta verze tuto plnou revizi vyžaduje a díky tomuto faktu je možné její okamžité zveřejnění v Google obchodě. Rozdílný je také maximální povolený počet uživatelů, u Alfa verze jde o 20 osob, u Beta verze o 200.[28] V následující podkapitole je popsán postup pro vydání Alfa verze. [29]

9.1 Postup

Pro vydání aplikace bylo potřeba vytvořit si vývojářský účet na Google Play Console, kde je registrační poplatek 25 dolarů. Android projekt je pomocí IDE Visual Studia archivován do podoby APK souboru. Následuje podepsání tohoto souboru klíčem vygenerovaným přes Java keytool za využití staženého Google Play certifikátu, viz obrázek 25a, a zaregistrování Google Play účtu, viz obrázek 25b. Takto podepsaný soubor je nyní možné nahrát do Google Play. Jak jde vidět na obrázku 27a, je ještě potřeba přidat název a provedené změny v tomto vydání. Jakmile dojde k tomuto prvnímu nahrání podepsané aplikace, dochází k odemknutí možnosti přímé distribuce do vývojářského účtu ve Visual Studiu, viz obrázek 26. Nyní zbývá vyčkat na kontrolu přidaného vydání, viditelnou v přehledu na obrázku 27b. Jedná se však o Alfa verzi, takže v tomto okamžiku může být aplikace testována. K tomuto účelu je potřeba pozvat uživatele s přidělenou rolí a stanovenými právy, viz obrázek 28. [30]



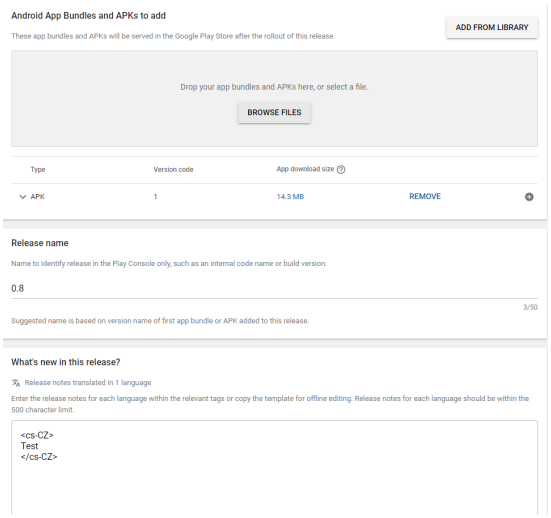
(a) Podepisování archivované aplikace za pomocí vygenerovaného klíče



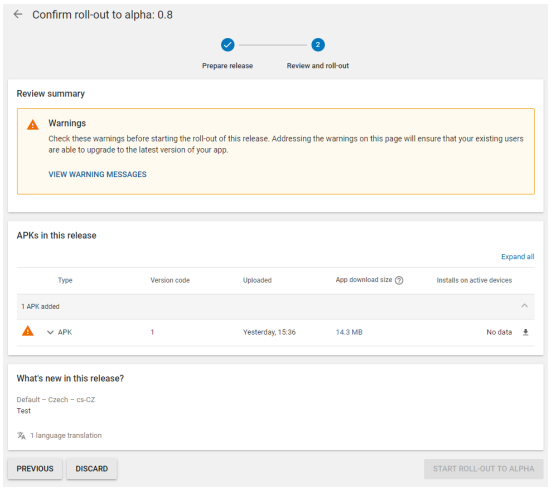
(b) Přidání Google Play účtu k projektu

Obrázek 25: Podepisování aplikace a přidání Google Play účtu

Obrázek 26: Přímé nahrávání APK souboru do Google Play

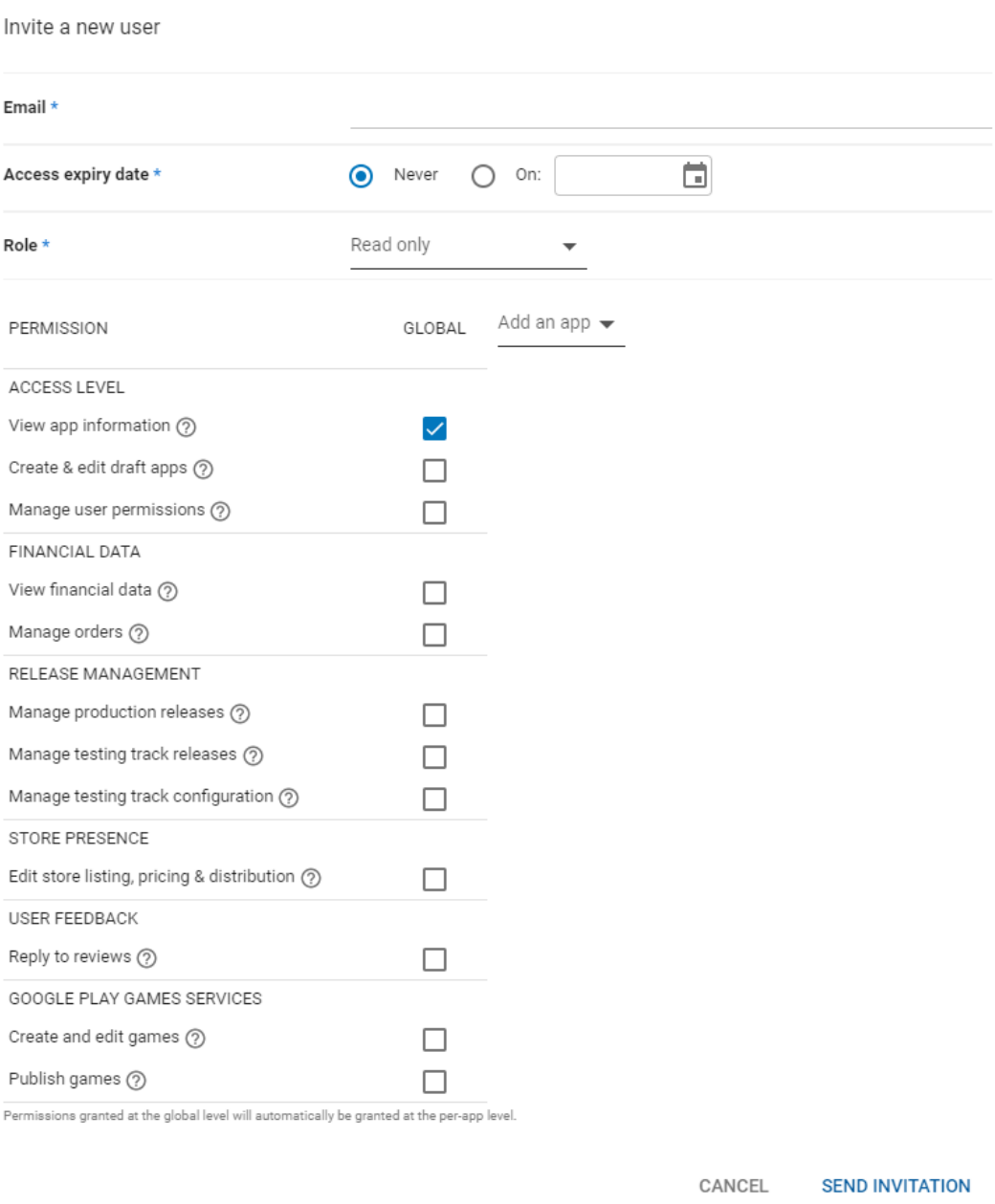


(a) Nahrání APK souboru do Google Play



(b) Přehled přidaného vydání

Obrázek 27: Nahrávání aplikace



Obrázek 28: Pozvání nového uživatele k testování

10 Uživatelské testování

Poslední část této práce se zabývá provedeným otestováním prototypu mobilní aplikace. Hlavní důraz byl kladen na testování přístupových práv pro jednotlivé role, vlastní funkčnost a zlepšení uživatelského rozhraní s cílem zlepšit přehlednost aplikace.

10.1 Testovací prostředí a způsob testování

Testování se zúčastnily celkem dva testovací subjekty s využitím následujících telefonů. Prvním z nich je Xiaomi Mi A2 s verzí Androidu 10. Druhým je Samsung S7 s verzí Androidu 8 - Oreo. Testování bylo prováděno formou zadání úkolu v podobě provedení specifické akce v aplikaci. Byl zapsán postup při vykonávání těchto úkolů a na jejich základě došlo k vyhodnocení a úpravám uživatelského rozhraní.

10.2 Zadané úkoly

1. Prvním úkolem bylo zaregistrování nového uživatelského účtu, přihlášení, přechod do celkového přehledu účtu a zde přidání nové připomínky s libovolným textem a datem.

2. Druhým úkolem bylo vytvoření nové pracovní skupiny s alespoň dvěma uživateli a v jejím rámci k vytvoření projektu s těmito členy vytvořené pracovní skupiny, poté ještě zadat libovolnému uživateli v projektu úkol. Bylo jedno, zda úkol zadá přímo přes stránku projektu, nebo přes všeobecný přehled.

3. Třetím a zároveň posledním úkolem bylo zobrazení informací o uživateli v rámci projektu a poté vyloučení tohoto člena z projektu. Nakonec mělo dojít i ke smazání celého vytvořeného projektu.

10.3 Provedení úkolu

1. První úkol proběhl bez nějakých vážnějších zádrhelů na straně obou testovacích subjektů. Menší komplikace byly zaznamenány akorát při snaze přejít do všeobecného přehledu, k čemuž je po přihlášení potřeba využít boční menu.

2. Ve druhém úkolu došlo díky znalosti celkového přehledu a zde existující záložky pracovních skupin k jejich rychlému a bezproblémovému založení. Do stránky pracovní skupiny byl přístup delší, ale nakonec se povedl a vzhledem k podobnosti mezi vytvářením projektů a pracovních skupin proběhlo jeho vytvoření také rychle. Při vytvoření úkolu oba testovací subjekty využily stránku celkového přehledu. To je odůvodněno absencí jakékoliv předchozí práce v projektu a navíc spatření záložky s úkoly v rámci celkového přehledu.

3. Při plnění třetího úkolu došlo k největším problémům, a to u obou uživatelů. U její první části chvíli trvalo přejít do členů projektu. Jakmile se však dostali na tuto stránku, kladně

ohodnotili přehlednost dostupných tlačítek a smazání člena skupiny proběhlo rychle. Hlavní zádrhel přišel v poslední části, tedy při smazání zvoleného projektu. Uživatel jedna vyzkoušel všechny stránky, které lze v projektu nalézt, od přehledu členů přes komentáře a přílohy. Až po vyzkoušení všech možností a ztrátě cenného času přišel na fakt, že na samotný projekt je potřeba kliknout a podržet ho, čímž vyskočí kontextová možnost smazání. Uživatel číslo dva prošel stránkou členů a komentářů a nakonec si raději vyžádal radu.

10.4 Vyhodnocení a provedené úpravy

Z provedeného testování je jasně patrné, že je nutné se podrobněji zaměřit na třetí úkol, který u jednoho subjektu zabral příliš mnoho času a u druhého skončil dokonce neúspěchem. Na základě dosažených zjištění bylo upuštěno od kontextových akcí při práci s jednotlivými položkami listu. Jako náhrada byla přidána tlačítka s ikonami, které byly uživateli hodnoceny pozitivně.

11 Závěr

Cílem této bakalářské práce bylo vytvořit mobilní aplikaci na platformě Xamarin.Forms s možností nasazení na platformách Android a iOS. Tato aplikace měla sloužit pro správu projektů a fungovat jako tlustý klient pro webovou aplikaci.

Úvodní část se v prvé řadě věnuje historickému vývoji, z něhož je patrné, že postupně vykrystalizovaly dva dominantní operační systémy Android a iOS, pro které je aplikace vyvíjena. Dále rozebírá všeobecné výhody a nevýhody, jež přináší multiplatformní vývoj oproti nativnímu. Poté jsou uvedeny jednotlivé přístupy k vývoji pro vícero platforem a popisu nejznámějších frameworků, reprezentující tyto směry. Nakonec je podrobněji popsán samotný Xamarin a Xamarin.Forms.

V návrhu aplikace jde o popis webové databáze, včetně jejího schématu. Jsou zde zahrnuty také knihovny a rozšíření, které jsou využívány. Kromě toho je také vysvětlen systém rolí jasně vymezující pravomoci a rámce. Pro lepší orientaci funkcionalit, spadající pod jednotlivé role, je navíc přidán Use case diagram. Některé ze složitějších případů užití jsou rozepsány v tabulkách pro jejich lepší pochopení.

Implementační část se zabývá jednotlivými funkcemi a popisem uživatelského rozhraní, u kterého byl kladen důraz na přehlednost a intuitivnost. Pro jejich zajištění bylo provedeno uživatelské testování s pevně stanovenými úkoly a vyhodnocením. Pro optimalizaci a lepší práci s aplikací byla využita lokální databáze. Důležitým požadavkem byla přítomnost upozornění na různé akce v systému, jež bylo implementováno specificky pro platformu Android. Aplikace byla s využitím popsaného postupu úspěšně nahrána na obchod Google Play v Alfa verzi.

O aplikaci během vývoje projevila zájem jedna firma. Pro přímé zavedení do ostrého provozu je potřeba přidat webové rozhraní, které nebylo součástí původní práce. Dále je nutné vylepšení zabezpečení komunikace mezi aplikací a webovou databází. Jako jedna z možností se nabízí využít autorizační OAuth 2.0 protokol s generací bezpečnostního tokenu.

Vypracování této práce mi přineslo nové poznatky a zkušenosti z oblasti multiplatformního vývoje. Vzhledem ke stále rostoucí oblibě tohoto přístupu k mobilnímu vývoji vnímám tyto nové znalosti jako žádané a dobře využitelné na pracovním trhu.

Zdroje

1. *Vývoj tržního podílu Androidu a iOS* [online] [cit. 2020-04-30]. Dostupné z: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.
2. *Dokumentace frameworku Apache Cordova* [online] [cit. 2020-04-30]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview/index.html.
3. *Dokumentace frameworku PhoneGap* [online] [cit. 2020-04-30]. Dostupné z: http://docs.phonegap.com/phonegap-build/overview/.
4. *Ukázka kódu ve frameworku Apache Cordova* [online] [cit. 2020-04-30]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/platforms/windows/index.html.
5. *Oficiální stránky frameworku React Native* [online] [cit. 2020-04-30]. Dostupné z: https://reactnative.dev/.
6. *Dokumentace k funkcionalitě Fast Refresh* [online] [cit. 2020-04-30]. Dostupné z: https://reactnative.dev/docs/fast-refresh.
7. *Ukázka kódu ve frameworku React Native* [online] [cit. 2020-04-30]. Dostupné z: https://zach.codes/autofocus-inputs-in-react-native/.
8. *GitHub frameworku NativeScript* [online] [cit. 2020-04-30]. Dostupné z: https://github.com/NativeScript/NativeScript.
9. *GitHub frameworku React Native* [online] [cit. 2020-04-30]. Dostupné z: https://github.com/facebook/react-native.
10. *Oficiální obchod pro framework NativeScript* [online] [cit. 2020-04-30]. Dostupné z: https://market.nativescript.org/.
11. *Společnosti využívající Flutter* [online] [cit. 2020-04-30]. Dostupné z: https://flutter.dev/showcase.
12. *Jazyk Dart* [online] [cit. 2020-04-30]. Dostupné z: https://dart.dev/.
13. *Dokumentace Dart VM* [online] [cit. 2020-04-30]. Dostupné z: https://mrale.ph/dartvm/.
14. *Ukázka kódu ve frameworku Flutter* [online] [cit. 2020-04-30]. Dostupné z: https://blog.codemagic.io/how-we-built-flutter-app-presented-at-mwc-19-in-one-month/.
15. *Srovnání tradičního Xamarinu se Xamarin.Forms* [online] [cit. 2020-04-30]. Dostupné z: https://docs.microsoft.com/xamarin.
16. *Dokumentace k jazyku XAML* [online] [cit. 2020-04-30]. Dostupné z: https://docs.microsoft.com/xamarin/xamarin-forms/xaml/.
17. *Mapování Button na nativní prvky* [online] [cit. 2020-04-30]. Dostupné z: https://docs.microsoft.com/xamarin.

18. *Co je to NuGet* [online] [cit. 2020-04-30]. Dostupné z: https://docs.microsoft.com/nuget/what-is-nuget.

19. *GitHub knihovny SQLite-net* [online] [cit. 2020-04-30]. Dostupné z: https://github.com/praeclarum/sqlite-net.

20. *NuGet knihovny Newtonsoft.Json* [online] [cit. 2020-04-30]. Dostupné z: https://www.nuget.org/packages/Newtonsoft.Json/.

21. *GitHub Rg.Plugins.Popup* [online] [cit. 2020-04-30]. Dostupné z: https://github.com/rotorgames/Rg.Plugins.Popup.

22. *Microsoft dokumentace Xamarin.Essentials* [online] [cit. 2020-04-30]. Dostupné z: https://docs.microsoft.com/xamarin/essentials/.

23. *GitHub PermissionsPlugin* [online] [cit. 2020-04-30]. Dostupné z: https://github.com/jamesmontemagno/PermissionsPlugin.

24. *GitHub knihovny Xamarin.Plugin.FilePicker* [online] [cit. 2020-04-30]. Dostupné z: https://github.com/jfversluis/FilePicker-Plugin-for-Xamarin-and-Windows.

25. *Azure Notification Hubs* [online] [cit. 2020-04-30]. Dostupné z: https://azure.microsoft.com/services/notification-hubs/.

26. *Jak připojit Firebase Cloud Messaging do Azure Notification Hubs* [online] [cit. 2020-04-30]. Dostupné z: https://docs.microsoft.com/azure/notification-hubs/notification-hubs-android-push-notification-google-fcm-get-started.

27. *Jak připojit Apple Push Notification service do Azure Notification Hubs* [online] [cit. 2020-04-30]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/notification-hubs/xamarin-notification-hubs-ios-push-notification-apns-get-started.

28. *Informace o verzích vydání na Google Play* [online] [cit. 2020-04-30]. Dostupné z: https://developers.google.com/assistant/console/releases.

29. *Informace pro vývojáře o Google Play* [online] [cit. 2020-04-30]. Dostupné z: https://developer.android.com/distribute/google-play.

30. *Vydání okamžité aplikace na Google Play* [online] [cit. 2020-04-30]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/7381861.